Redes Neurais Artificiais: Uma Introdução Prática

Prof. Dr. Guilherme de Alencar Barreto

Depto. Engenharia de Teleinformática (*DETI/UFC*)

URL: www.deti.ufc.br/~guilherme

Email: gbarreto@ufc.br

Março/2014

Ementa

1. Funcionalidades do neurônio biológico
2. Neurônio artificial de McCulloch-Pitts (MP)
3. Análise geométrica do neurônio MP
4. Portas lógicas AND, OR e NOT
5. Rede Perceptron Simples (PS) e aplicações
6. Problemas não-linearmente separáveis e a porta lógica XOR
7. Implementação da porta lógica XOR via redes multicamadas
8. Rede Perceptron Multicamadas (MLP)
9. Algoritmo de retropropagação do erro (*error backpropagation*)
10. Dicas de treinamento, teste e validação da rede MLP

Material Didático

1. Notas de aula em PDF.

2. Principe, J. C., Euliano, N. R. & Levebvre, W. C. (2000). **Neural and Adaptive Systems: Fundamentals through Simulations**, 1a. Edição, John Wiley and Sons.

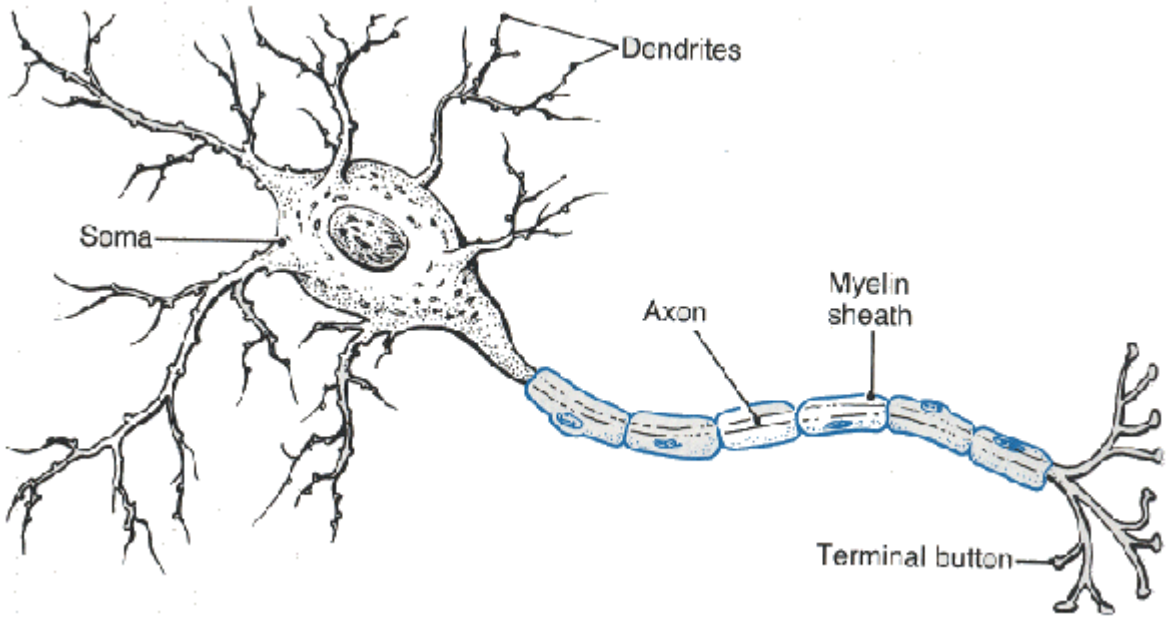
3. Apostila de Redes Neurais, PPGETI-DETI-UFC.

4. Tutorial do Software *NeuroSolutions*.

1. O Neurônio Biológico

O neurônio é antes de tudo uma célula, mas uma célula especial.

Partes: (*i*) dendritos, (*ii*) sinapses, (*iii*) corpo celular e (*iv*) axônio



1. O Neurônio Biológico

(*i*) **dendritos** – Ramificações correspondentes aos canais de entrada de informação (sinais elétricos, escala mVolts).

(*ii*) **sinapses** – Pontos de contato entre neurônios onde há passagem de neurotransmissores do axônio de um neurônio para os dendritos de outro neurônio.

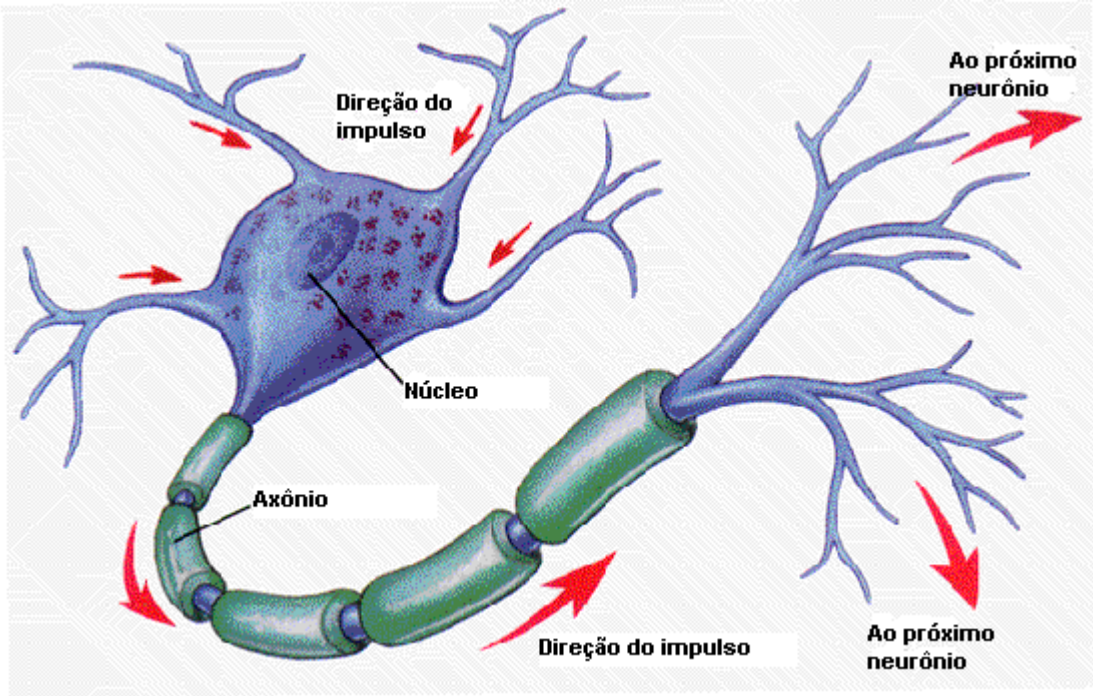
(*iii*) **corpo celular –** Local onde é feito o balanço energético da célula nervosa (soma das contribuições de energia).

(*iv*) **Axônio** – Canal de saída do neurônio, ou seja, caminho de propagação dos impulsos nervosos em direção a outros neurônios ou músculos.

1. O Neurônio Biológico

O fluxo da informação ocorre sempre no sentido:

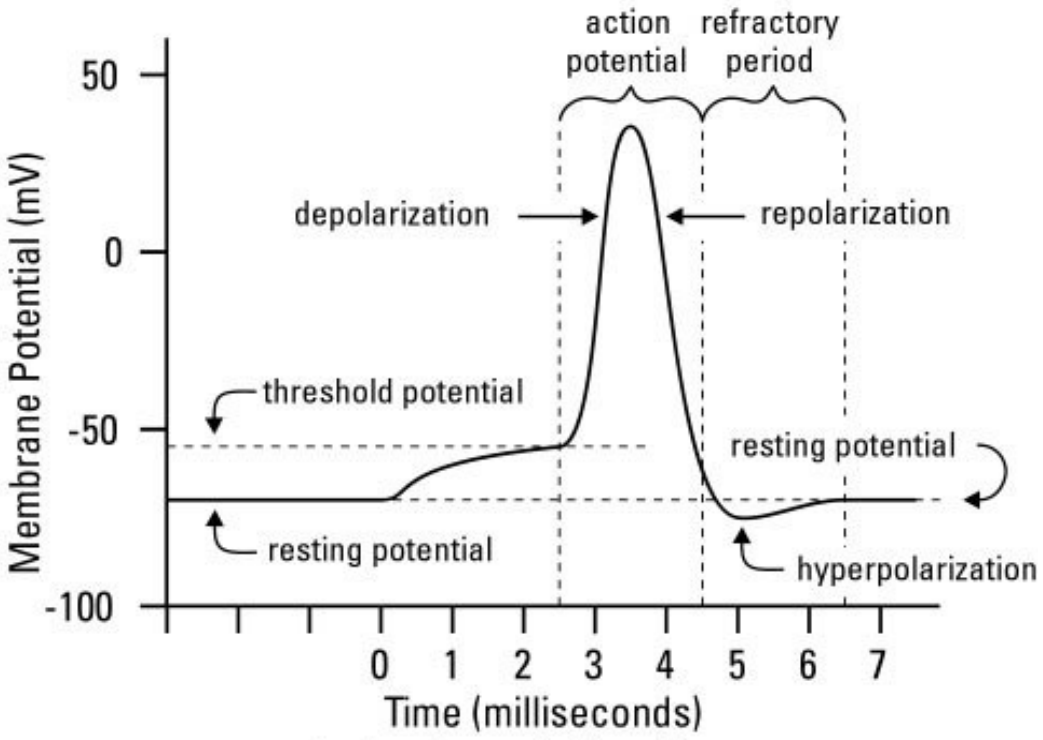
Dendritos ➔ Corpo Celular s ➔ Axônio



1. O Neurônio Biológico

O axônio emite um impulso elétrico (potencial de ação) apenas se o balanço energético realizado no corpo celular for maior que um certo limiar. Neste caso, diz-se que o neurônio **disparou** ou está **ativado**.

Potencial de repouso: -70mV
Limiar de disparo: -55 mV

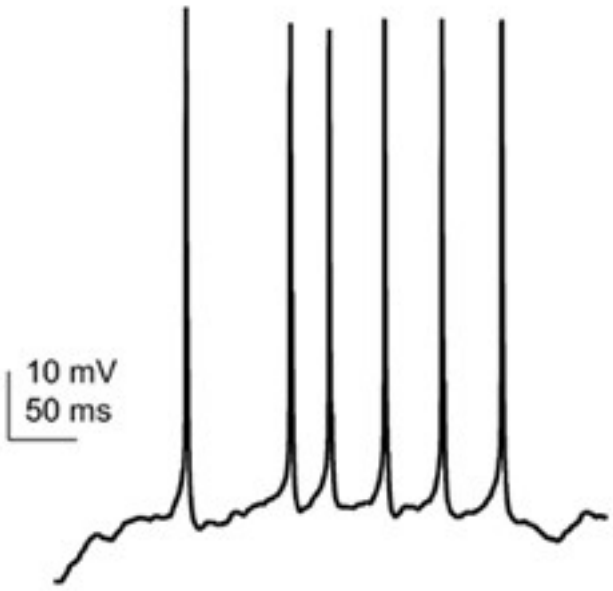


Action Potential in a Neuron

1. O Neurônio Biológico

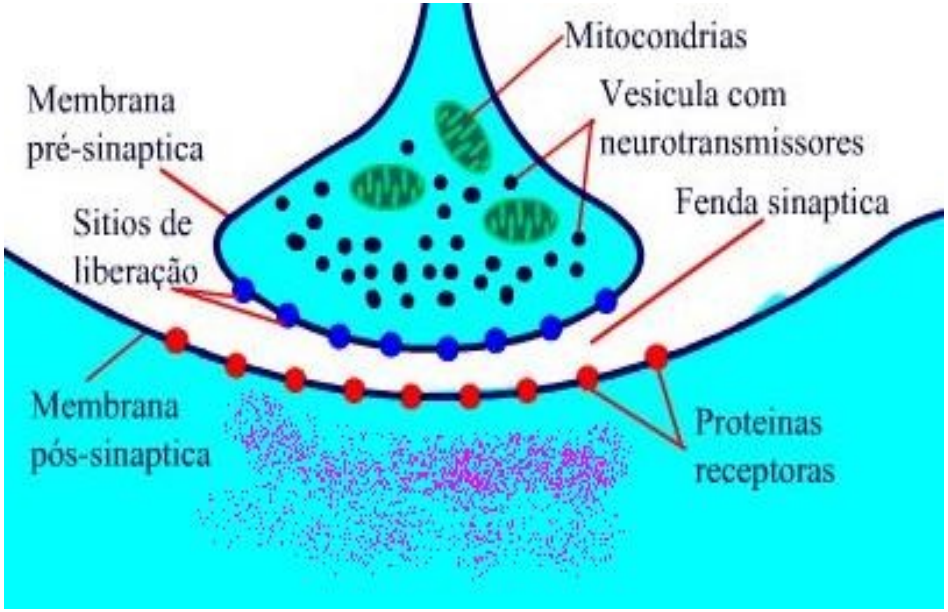
Um neurônio devidamente estimulado emite um trem de potenciais de ação ao longo de seu axônio.

A informação é então codificada na freqüência dos potenciais de ação!



1. O Neurônio Biológico

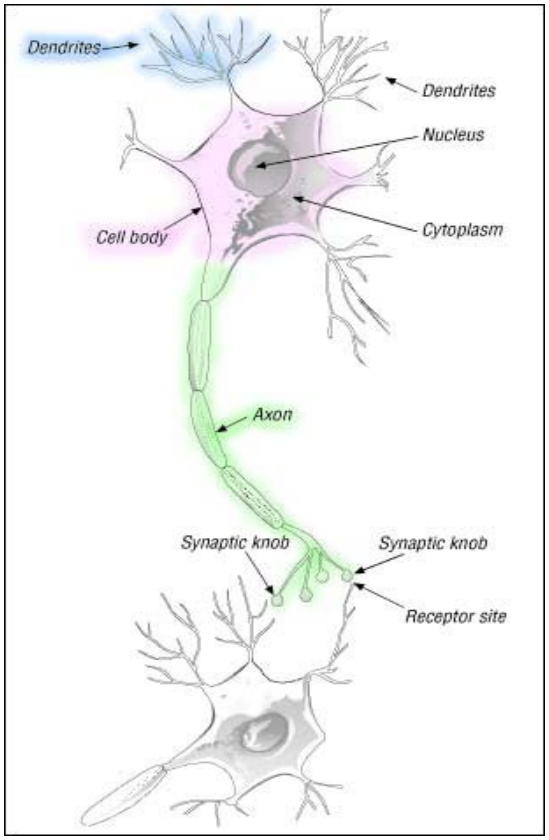
A chegada de um trem de pulso no botão sináptico localizado na região terminal do axônio provoca a liberação de transmissores na fenda sináptica.



Sinapses podem ser *excitatórias* (facilitam a passagem do potencial de ação) ou inibitórias (inibem a passagem do potencial de ação).

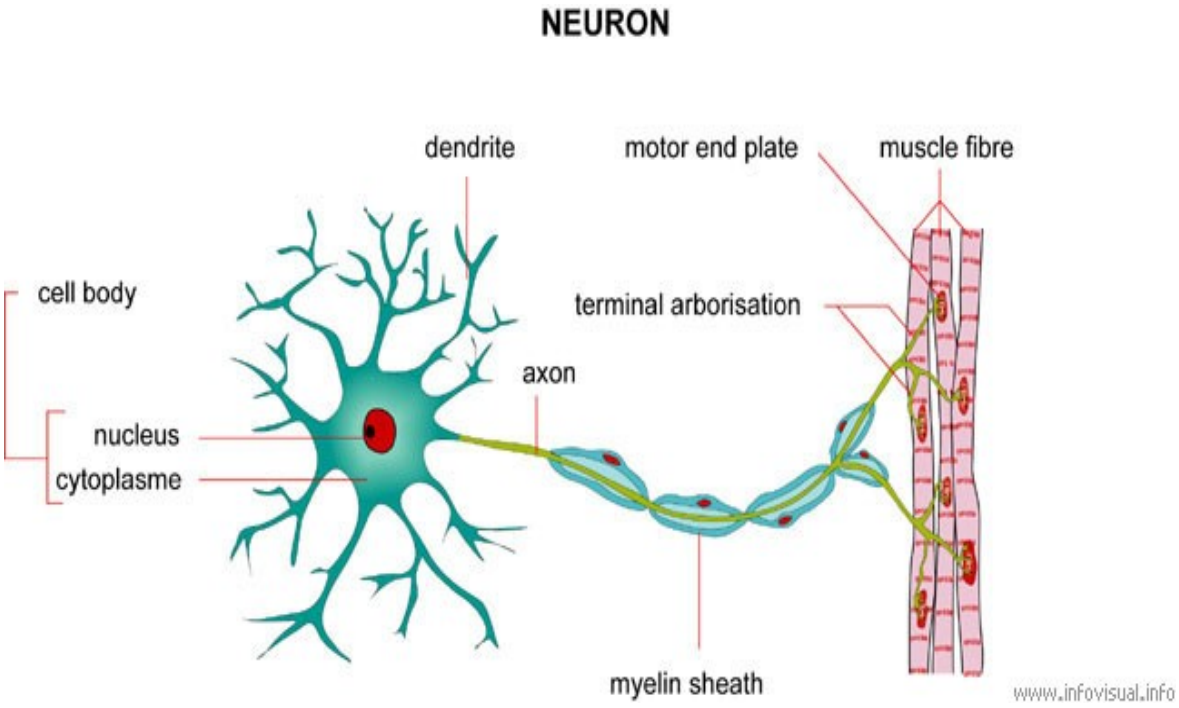
1. O Neurônio Biológico

Neurônios podem se conectar com outros neurônios ...



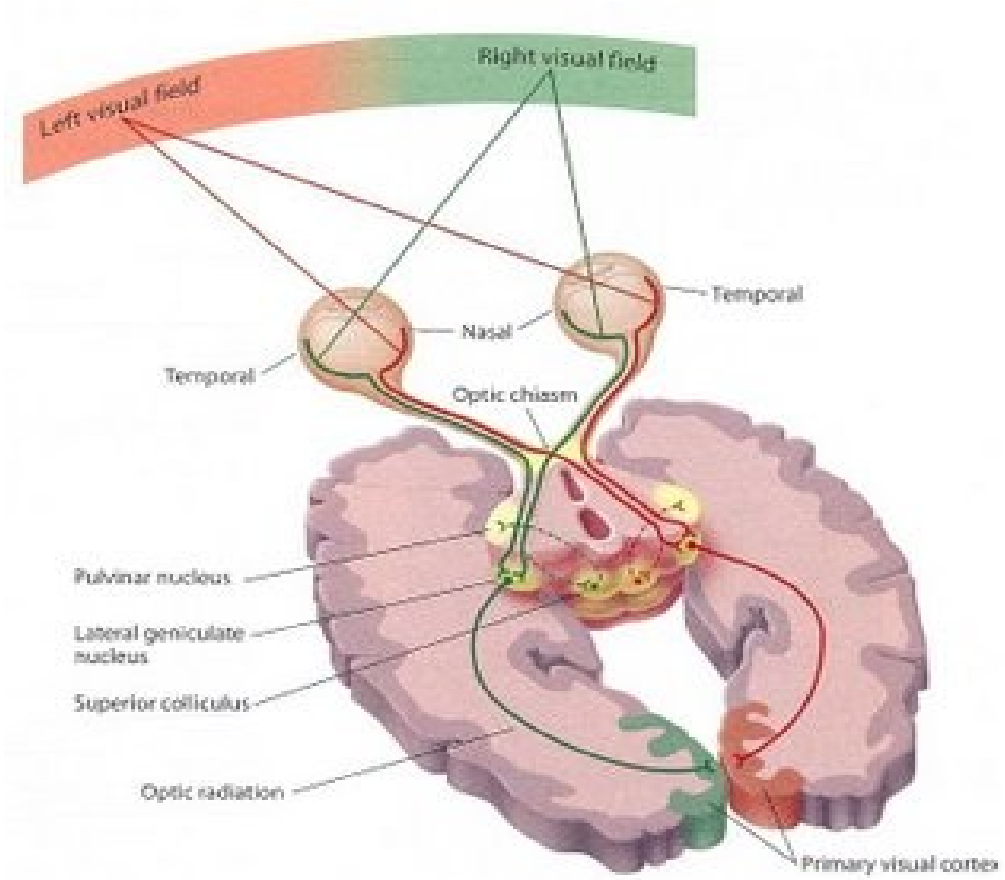
1. O Neurônio Biológico

... com os músculos diretamente ...



1. O Neurônio Biológico

... ou com orgãos sensoriais (e.g. visão).



1. O Neurônio Biológico

Fatos Curiosos

Há cerca de 100 bilhões deles no cérebro e na coluna vertebral.

Cada neurônio tem cerca de 10.000 sinapses com outros neurônios.

A maioria deles está localizado no córtex cerebral.

O córtex existe apenas nos cérebros de mamíferos.

O córtex é identificado popularmente como massa cinzenta.

O córtex é a estrutura responsável pelas habilidades cognitivas superiores, tais como memória, raciocínio lógico, linguagem, consciência, dentre outras.

1. O Neurônio Biológico

Mais Fatos Curiosos

O cérebro produz sim **novos** neurônios (e.g. hipocampo).

O tempo de propagação de um impulso no axônio é da ordem de milissegundos!

milissegundos????

Logo, a freqüência de disparo de um neurônio é da ordem de kHz!

Como pode um elemento tão lento, executar tarefas tão rápido???

1. O Neurônio Biológico

Consumo Energético do Cérebro Humano

O peso do cérebro é aprox. 2% do peso de uma pessoa.

Mesmo em repouso, o cérebro consome 20% de sua energia.

Assim, se o consumo médio de energia de um adulto típico é de 100W. Então o cérebro consome em média 20W.

O cérebro consome 10 vezes mais energia que o resto do corpo por grama de tecido.

Ementa

1. Funcionalidades do neurônio biológico
2. **Neurônio artificial de McCulloch-Pitts (MP)**
3. Análise geométrica do neurônio MP
4. Portas lógicas AND, OR e NOT
5. Rede Perceptron Simples (PS) e aplicações
6. Problemas não-linearmente separáveis e a porta lógica XOR
7. Implementação da porta lógica XOR via redes multicamadas
8. Rede Perceptron Multicamadas (MLP)
9. Algoritmo de retropropagação do erro (*error backpropagation*)
10. Dicas de treinamento, teste e validação da rede MLP

2. Neurônio de McCulloch-Pitts

Modelo matemático de um neurônio biológico proposto em

W. S. McCullogh and W. Pitts (1943). "A logical calculus of the ideas immanent in nervous activity", *Bulletin of Mathematical Biophysics*, vol. 5, p. 115-133.

É bom lembrar que todo modelo é apenas uma aproximação do fenômeno ou objeto real cujas funcionalidades se pretende estudar.

"All models are wrong, but some are useful."

George E. P. Box

Assim, o neurônio M-P é uma aproximação útil do neurônio real, pois serve até hoje como bloco construtivo básico de algoritmos de redes neurais.

2. Neurônio de McCulloch-Pitts

Na construção do neurônio M-P se está interessado em modelar aspectos ligados ao **Processamento da Informação** em um neurônio biológico.

Entende-se por processamento da informação os caminhos e etapas pelas quais passam os potenciais de ação que trafegam de

(*i*) um neurônio a outro neurônio,

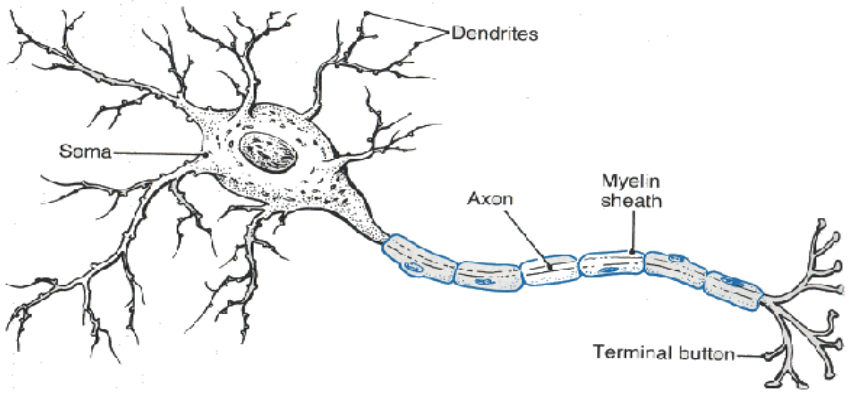
(*ii*) receptores sensoriais a um neurônio, ou

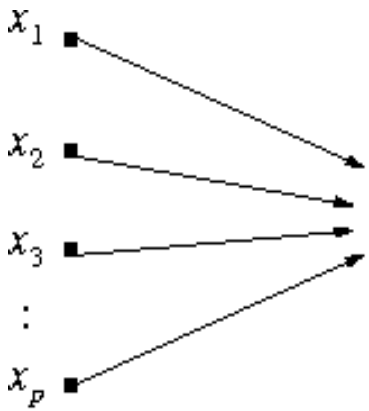
(*iii*) de um neurônio a um atuador (e.g. músculo).

Assim, devemos desenvolver modelos matemáticos que representem os **dendritos**, as **sinapses**, o **corpo celular** e o **axônio**.

2. Neurônio de McCulloch-Pitts

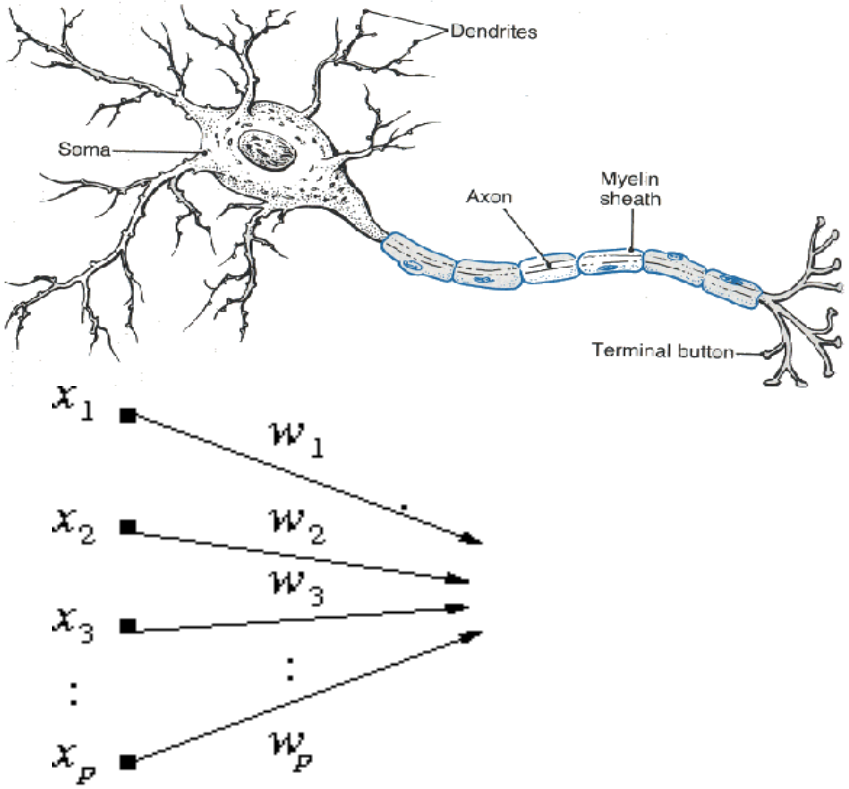
Passo 1: Cada ramo da árvore dendrítica é modelado como uma linha ou canal de transmissão por onde flui a informação de entrada (x_j, j=1, ..., p).





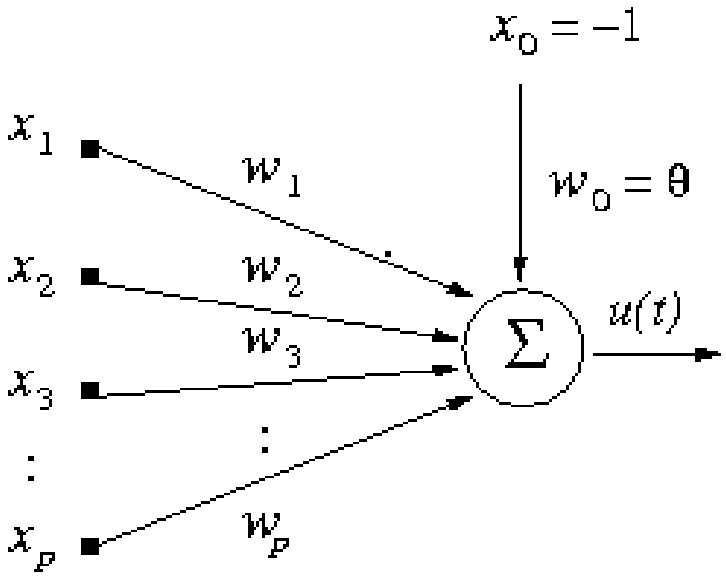
2. Neurônio de McCulloch-Pitts

Passo 2: A força (ou eficiência) das conexões sinápticas de uma certa árvore dendrítica é modelada como um fator (peso sináptico), cujo papel é modular o fluxo de sinais passando por uma certa árvore dendrítica.



2. Neurônio de McCulloch-Pitts

Passo 3: A função do corpo celular de realizar o balanço ou acúmulo energético é modelada por uma operação de somatório sobre as entradas moduladas pelos pesos sinápticos.



$$u = w_1x_1 + w_2x_2 + ... + w_px_p - \theta$$

x_1, x_2: entradas

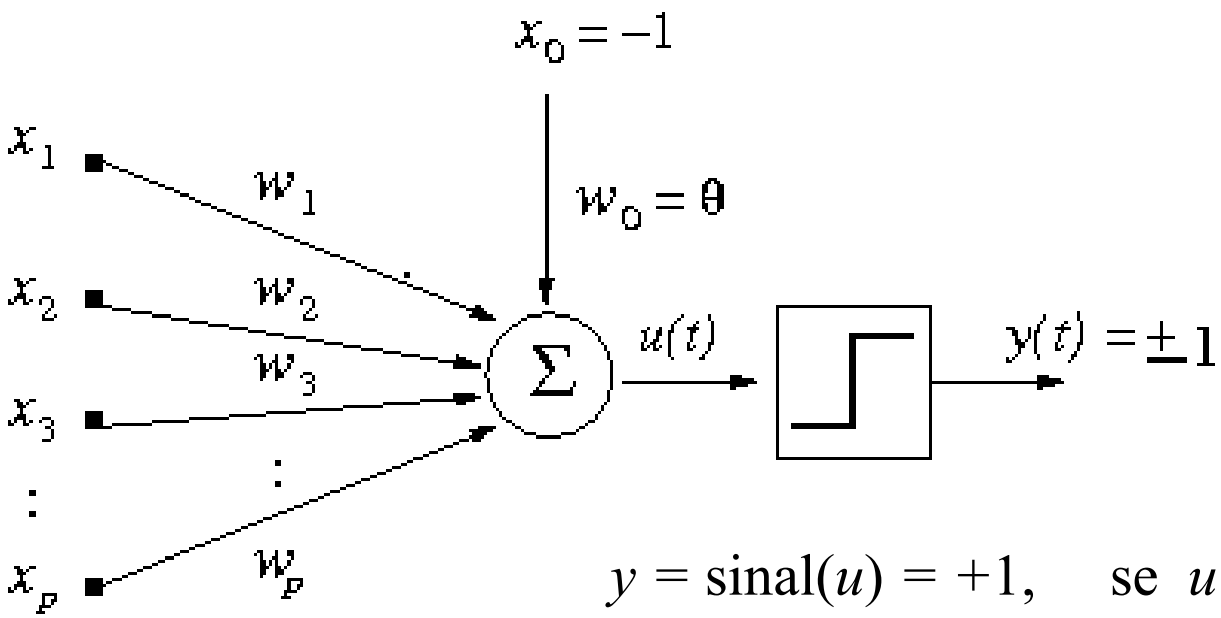
w_1, w_2 : pesos sinápticos

θ: limiar (*bias*)

u : ativação

2. Neurônio de McCulloch-Pitts

Passo 4: O axônio é modelado como uma chave ON-OFF, que indica se o neurônio respondeu ao estímulo atual. Em outras palavras, se houve ou não o envio de um potencial de ação.

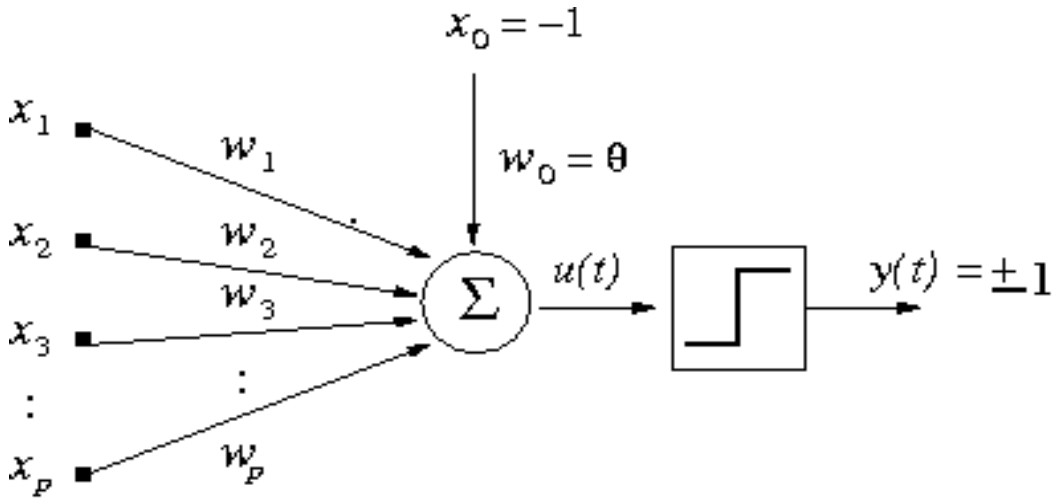


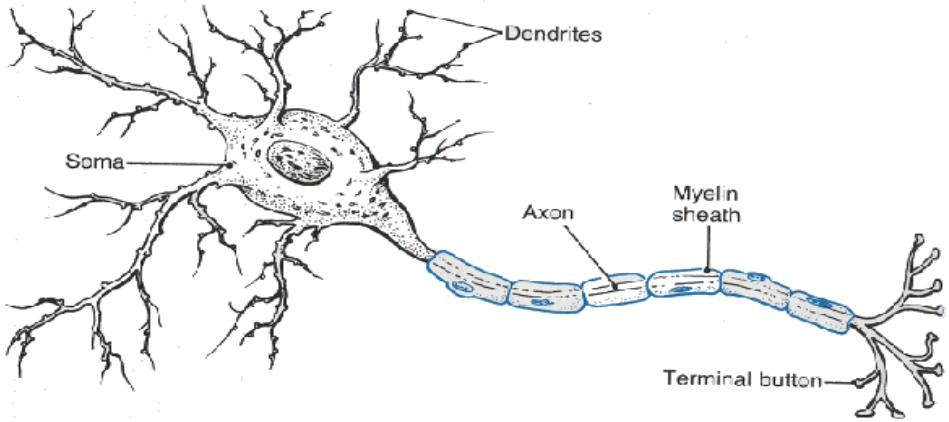
$y = \text{sinal}(u) = +1, \quad \text{se } u > 0$

$y = \text{sinal}(u) = -1, \quad \text{caso contrário.}$

2. Neurônio de McCulloch-Pitts

Modelo Completo do Neurônio Artificial de McCulloch-Pitts



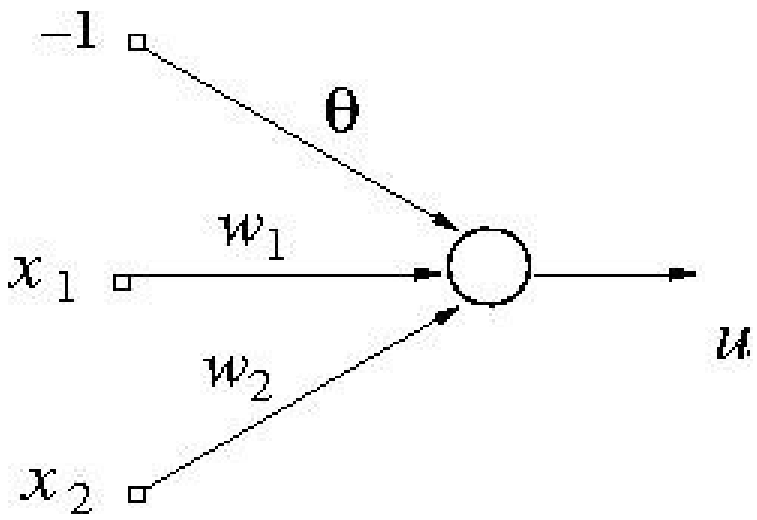


Ementa

1. Funcionalidades do neurônio biológico
2. Neurônio artificial de McCulloch-Pitts (MP)
3. **Análise geométrica do neurônio MP**
4. Portas lógicas AND, OR e NOT
5. Rede Perceptron Simples (PS) e aplicações
6. Problemas não-linearmente separáveis e a porta lógica XOR
7. Implementação da porta lógica XOR via redes multicamadas
8. Rede Perceptron Multicamadas (MLP)
9. Algoritmo de retropropagação do erro (*error backpropagation*)
10. Dicas de treinamento, teste e validação da rede MLP

3. Análise Geométrica

Seja o neurônio artificial mostrado na figura abaixo.



x_1, x_2: entradas

w_1, w_2 : pesos sinápticos

θ: limiar (*bias*)

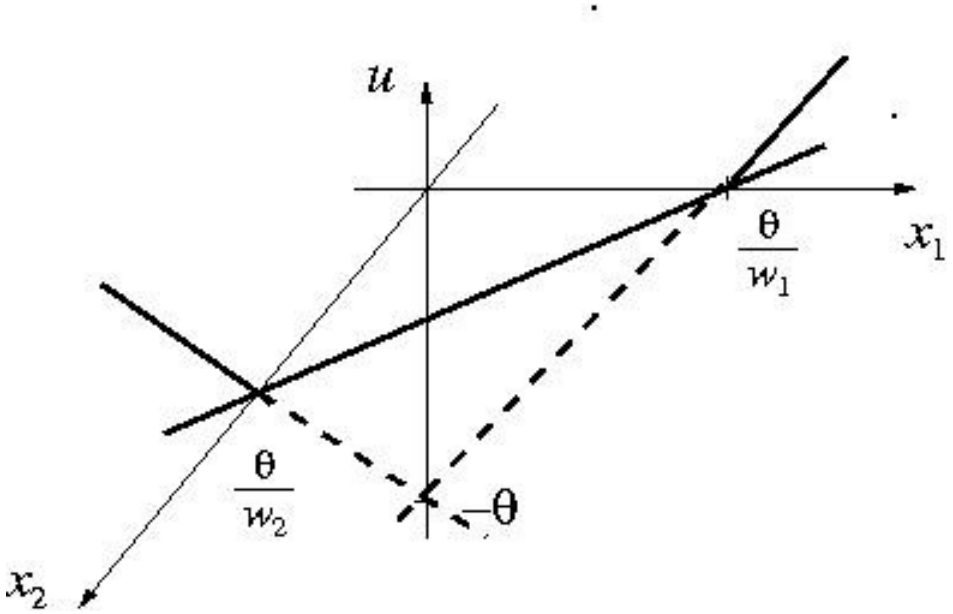
u : ativação

A ativação (u) do neurônio é dada por:

$$u = w_1x_1 + w_2x_2 - \theta \qquad (1)$$

3. Análise Geométrica

A Eq. (1) define um plano em (x_1, x_2, u).



Obs: O tracejado indica onde o plano está abaixo do plano (x_1, x_2).

3. Análise Geométrica

Para fins de classificação basta trabalhar no plano (x_1, x_2).

Isto equivale a fazer u=0 na equação do plano, ou seja:

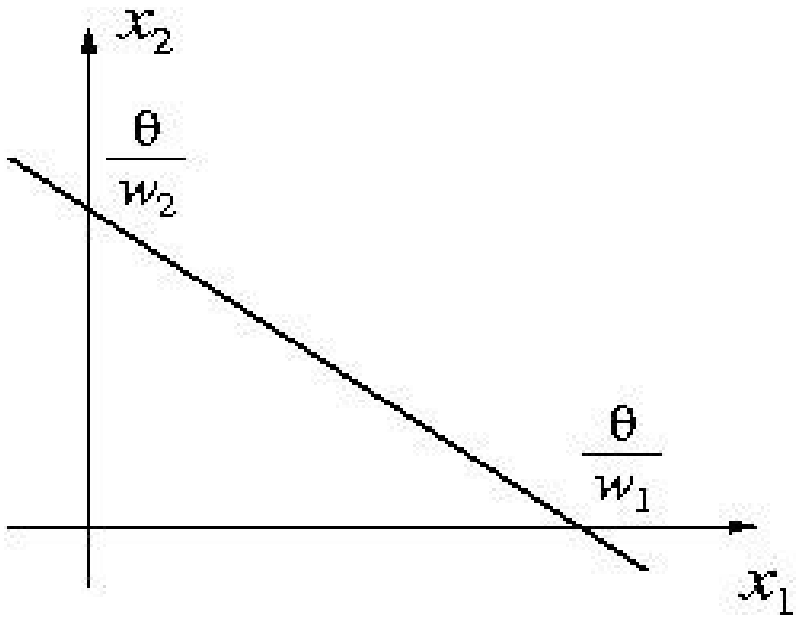
$$u = w_1 x_1 + w_2 x_2 - \theta = 0$$

Assim, a equação da reta no plano (x_1, x_2) é dada por:

$$x_2 = -(w_1/w_2)x_1 + \theta/w_2 \qquad (2)$$

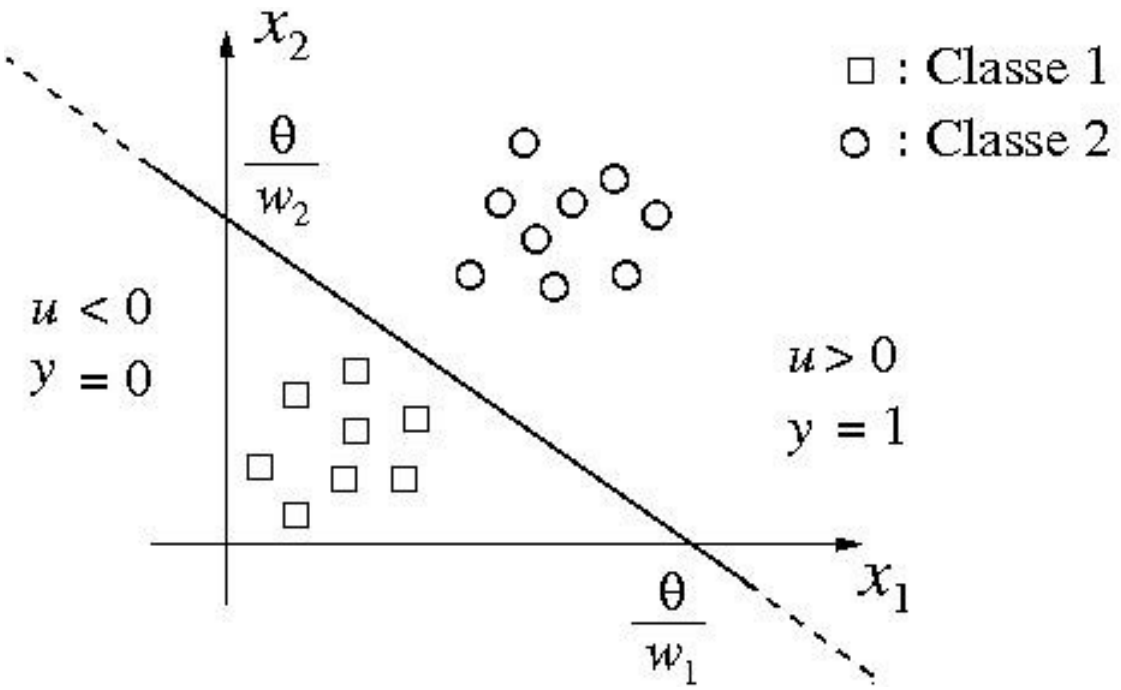
3. Análise Geométrica

A Eq. (2) define a seguinte reta em (x_1, x_2).



3. Análise Geométrica

Assim, um neurônio pode ser usado para separar com eficiência duas classes que estejam bem isoladas uma da outra.



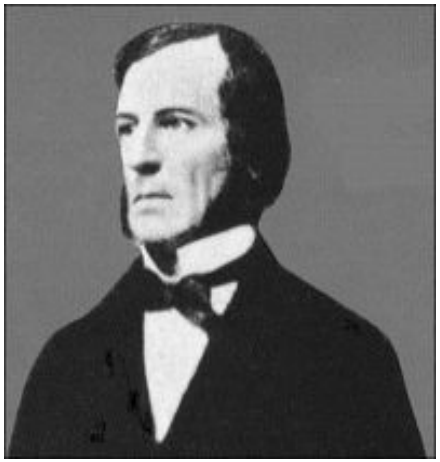
Ementa

1. Funcionalidades do neurônio biológico
2. Neurônio artificial de McCulloch-Pitts (MP)
3. Análise geométrica do neurônio MP
4. **Portas lógicas AND, OR e NOT**
5. Rede Perceptron Simples (PS) e aplicações
6. Problemas não-linearmente separáveis e a porta lógica XOR
7. Implementação da porta lógica XOR via redes multicamadas
8. Rede Perceptron Multicamadas (MLP)
9. Algoritmo de retropropagação do erro (*error backpropagation*)
10. Dicas de treinamento, teste e validação da rede MLP

4. Portas Lógicas And, Or e Not

Qual a relação entre portas lógicas e Inteligência Artificial?

George Boole (1854). "*An investigation into the Laws of Thought, on Which are Founded the Mathematical Theories of Logic and Probabilities*".



George Boole (02/11/1815 - 08/12/1864). Matemático e filósofo britânico. É o criador da Álgebra Booleana, base da atual aritmética computacional.

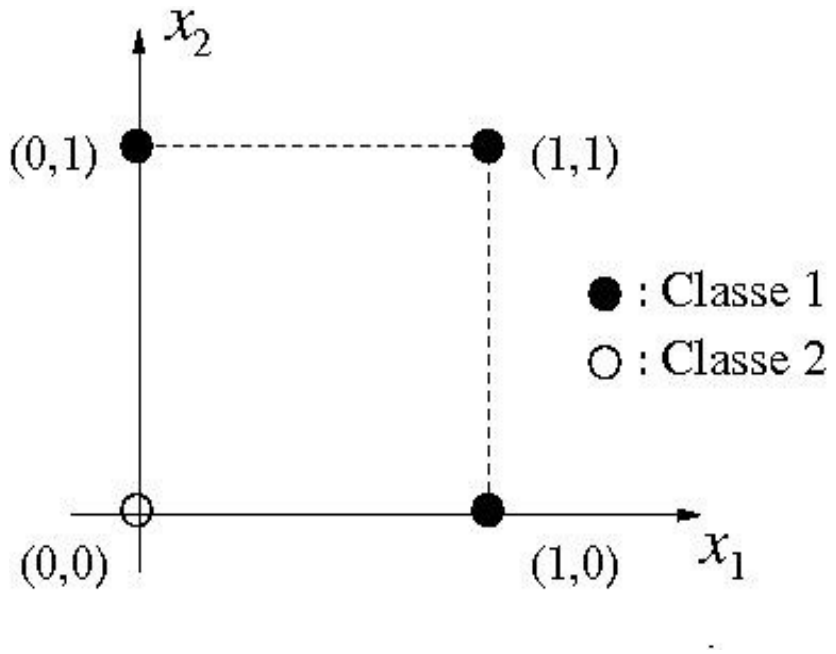
4. Portas Lógicas And, Or e Not

***Exemplo* 1**: Implementando funções lógicas (AND, OR, NOT).

Representação do Problema (Função OR)

Porta OR

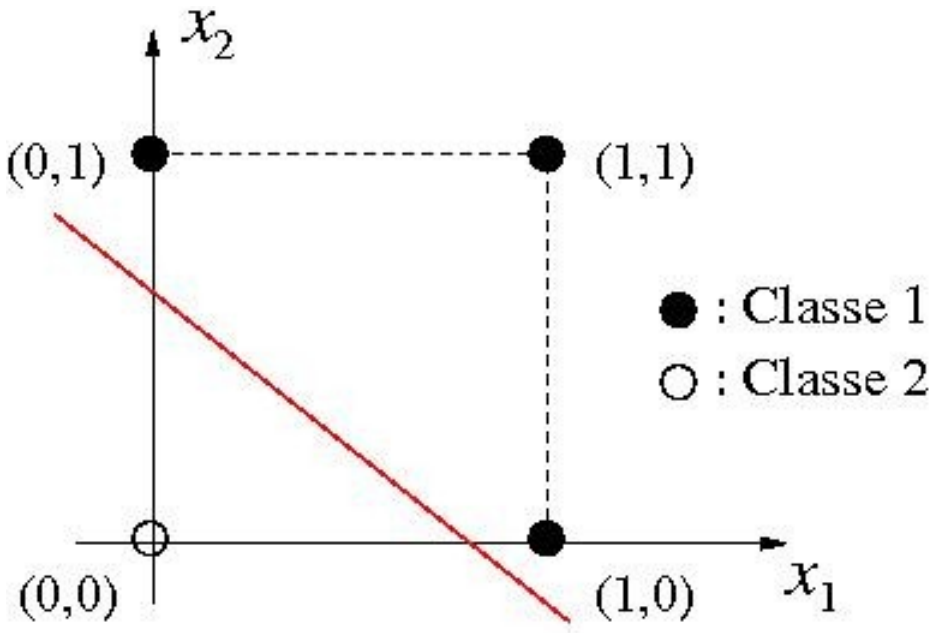
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1



4. Portas Lógicas And, Or e Not

Exemplo* 1 *(cont.): É possível encontrar uma reta que separe os pontos da Classe 1 (y=1) dos da Classe 2 (y=0)?

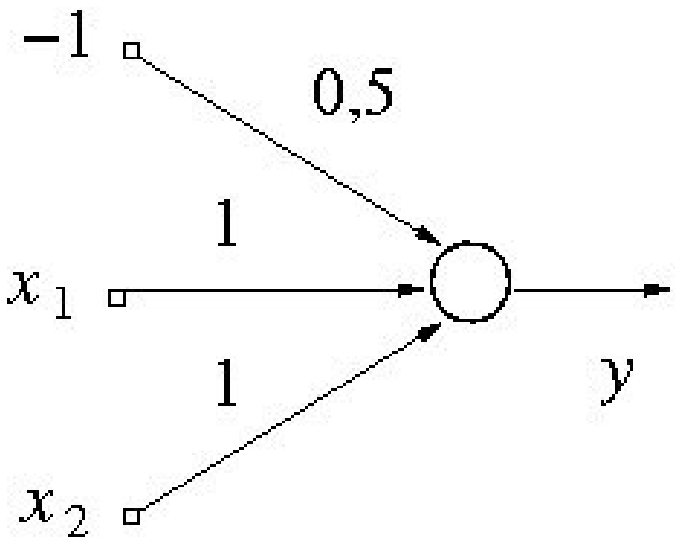
Resposta:
SIM!



Obs: Na verdade, é possível encontrar *infinitas* retas que separam as duas classes!

4. Portas Lógicas And, Or e Not

Exemplo 2 : O seguinte neurônio implementa a porta OR.



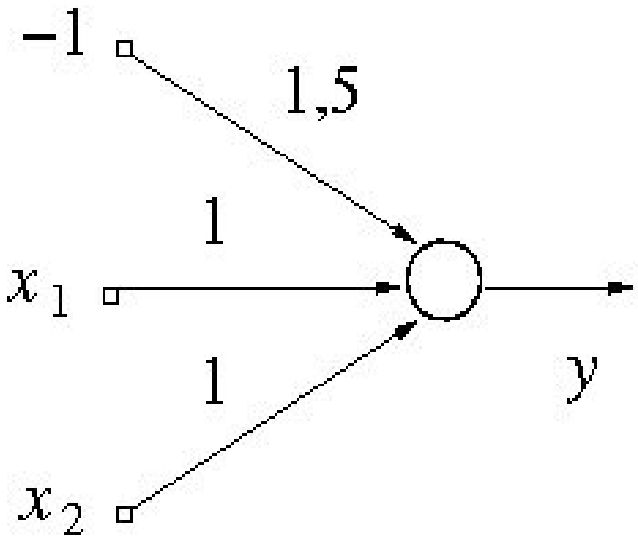
$w_1 = w_2 = 1$ e $\theta = 0{,}5$

$y = 1$, se $u \geqslant 0$.

$y = 0$, se $u < 0$.

4. Portas Lógicas And, Or e Not

***Exemplo* 3** : O seguinte neurônio implementa a porta AND.



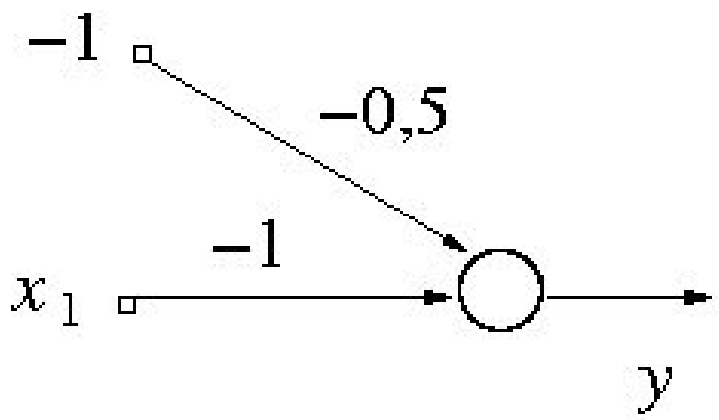
$w_1 = w_2 = 1$ e $\theta = 1{,}5$

$y = 1$, se $u \geqslant 0$.

$y = 0$, se $u < 0$.

4. Portas Lógicas And, Or e Not

***Exemplo* 4** : O seguinte neurônio implementa a porta NOT.



$w_1 = -1$ e $\theta = -0{,}5$

$y = 1$, se $u \geqslant 0$.

$y = 0$, se $u < 0$.

4. Portas Lógicas And, Or e Not

IMPORTANTE 1

O neurônio MP pode ser usado para implementar as portas lógicas AND, OR e NOT porque estas, do ponto de vista geométrico, podem ser interpretadas como um problema de classificação binária (duas categorias).

4. Portas Lógicas And, Or e Not

IMPORTANTE 2

O neurônio MP, do ponto de vista geométrico, pode ser intepretado como uma reta (2D), ou um plano (3D) ou ainda um hiperplano (> 3D), que é usado para separar duas categorias de dados distintas.

4. Portas Lógicas And, Or e Not

IMPORTANTE 3

Na implementação das portas lógicas AND, OR e NOT, os valores dos pesos e do limiar foram determinados pelo projetista com base na análise geométrica do problema.

Como fazer com que o neurônio M-P determine de forma automática os valores dos pesos e do limiar para um problema específico?

4. Portas Lógicas And, Or e Not

IMPORTANTE 4

Para que o neurônio M-P seja capaz de aprender sozinho a resolver um problema de classificação é necessário dotá-lo de uma regra de aprendizagem.

Uma regra de aprendizagem nada mais é do que uma equação que altera os valores dos pesos e do limiar em função dos erros cometidos durante a execução da tarefa de classificação.

4. Portas Lógicas And, Or e Not

Memorex 1

O neurônio M-P é um modelo simplificado do neurônio real.

O neurônio M-P possui p variáveis de entrada: $x_1, x_2, ..., x_p$

O neurônio M-P possui p pesos sinápticos: $w_1, w_2, ..., w_p$

O neurônio M-P possui um limiar de ativação: θ

O neurônio M-P possui uma variável de ativação: u

$$u = w_1x_1 + w_2x_2 + ... + w_px_p - \theta$$

O neurônio M-P possui uma variável de saída: y

$$y = \text{sinal}(u) = +1 \text{ (se } u > 0) \text{ ou } -1 \text{ (se } u \leq 0)$$

4. Portas Lógicas And, Or e Not

Memorex 2

O neurônio M-P nada mais é do que uma chave ON/OFF.

O neurônio M-P pode ser utilizado para implementar portas lógicas AND, OR e NOT.

As condições que definem a ativação (y=1) ou não (y=0 ou -1) do neurônio depende dos valores dos pesos e do limiar.

Assim, pode ser utilizado em problemas de reconhecimento de padrões que envolvam duas categorias (binários).

Ementa

1. Funcionalidades do neurônio biológico
2. Neurônio artificial de McCulloch-Pitts (MP)
3. Análise geométrica do neurônio MP
4. Portas lógicas AND, OR e NOT
5. **Rede Perceptron Simples (PS) e aplicações**
6. Problemas não-linearmente separáveis e a porta lógica XOR
7. Implementação da porta lógica XOR via redes multicamadas
8. Rede Perceptron Multicamadas (MLP)
9. Algoritmo de retropropagação do erro (*error backpropagation*)
10. Dicas de treinamento, teste e validação da rede MLP

5. Rede Perceptron Simples

A rede Perceptron Simples (PS) é considerada o primeiro algoritmo de redes neurais artificiais.

A rede PS foi proposta por Frank Rosenblatt em 1958.

F. Rosenblatt (1958). "The Perceptron: A probabilistic model for information storage and organization in the brain", *Psychological Review*, vol. 65, p. 386-408.

Perceptron Simples = Neurônio de M-P + Regra de Aprendizagem

A regra de aprendizagem é o mecanismo que torna a rede Perceptron Simples um dispositivo inteligente!

5. Rede Perceptron Simples

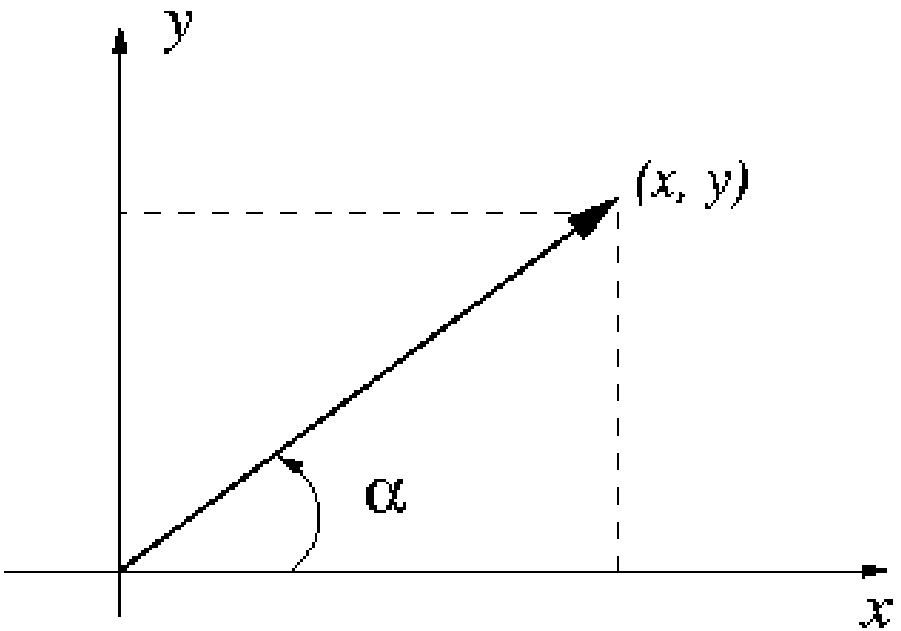
Para facilitar a análise as variáveis de entrada e os pesos vão ser representados como vetores de agora em diante.

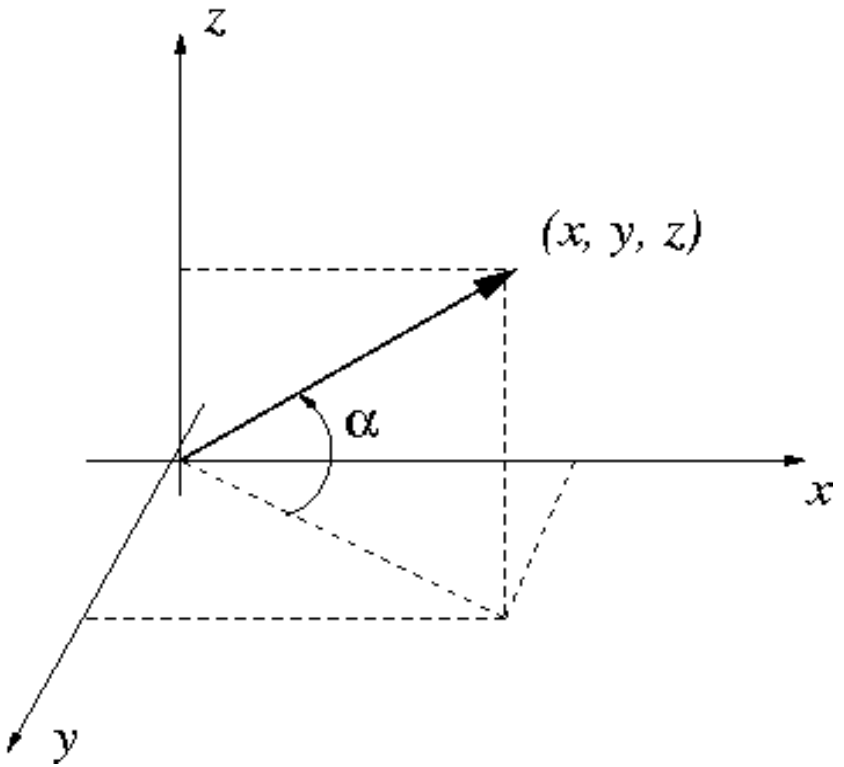
$$\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_p \end{pmatrix} = \begin{pmatrix} -1 \\ x_1 \\ \vdots \\ x_p \end{pmatrix} \qquad \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_p \end{pmatrix} = \begin{pmatrix} \theta \\ w_1 \\ \vdots \\ w_p \end{pmatrix}$$

5. Rede Perceptron Simples

Métodos Geométricos

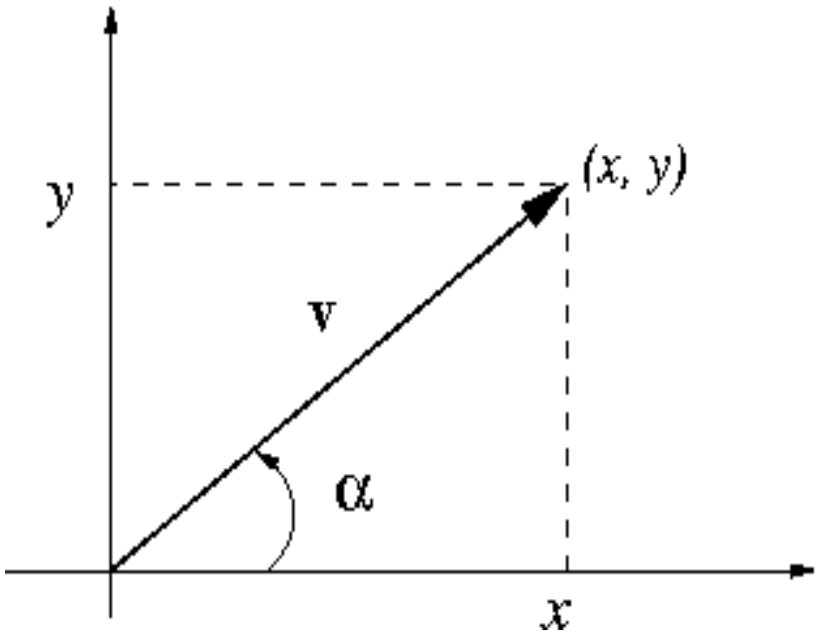
Um vetor é uma coordenada (ponto) em um espaço de dimensão *p*.





5. Rede Perceptron Simples

Comprimento de um Vetor em 2D



$$\|\mathbf{v}\| = \sqrt{x^2 + y^2}$$

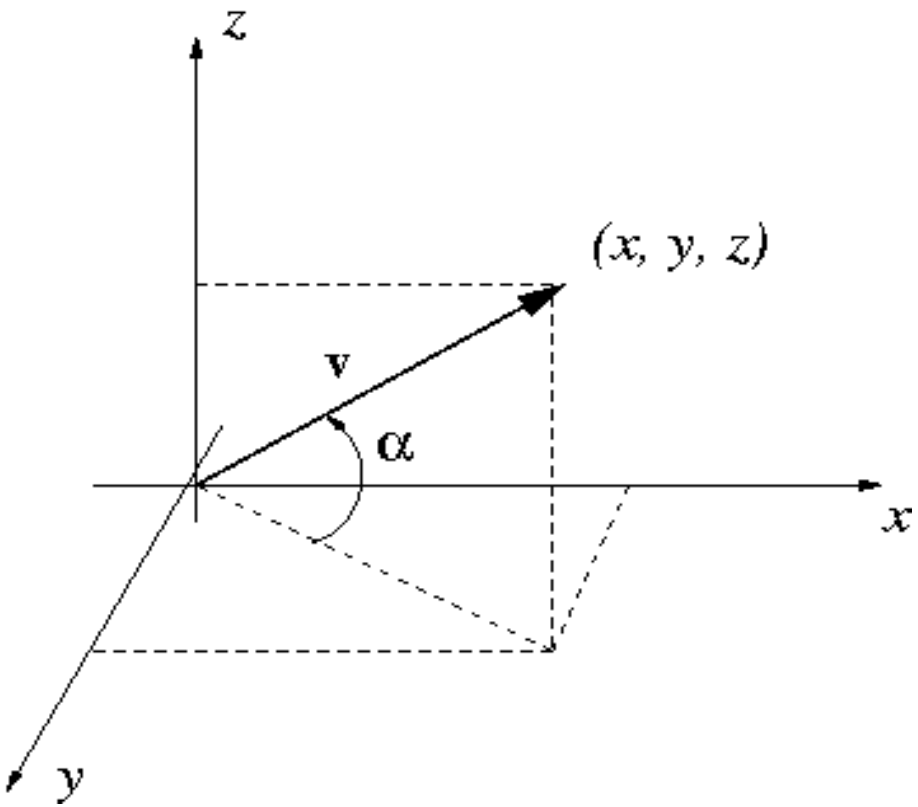
$$x = \|\mathbf{v}\| . \cos(\alpha)$$

$$y = \|\mathbf{v}\| . \sin(\alpha)$$

OBS: Usar teorema de Pitágoras!

5. Rede Perceptron Simples

Comprimento de um Vetor em 3D



$$\|\mathbf{v}\| = \sqrt{x^2 + y^2 + z^2}$$

OBS: Usar teorema de Pitágoras duas vezes!

5. Rede Perceptron Simples

Produto Escalar entre 2 Vetores

Definição 1:

$$u = \mathbf{w}^T \mathbf{x} = \mathbf{x}^T \mathbf{w}$$

$$= w_0 x_0 + w_1 x_1 + \ldots + w_p x_p$$

O produto escalar é definido como o produto de um vetor-linha por um vetor-coluna, o que equivale a multiplicar cada componente de um vetor pelo seu correpondente no outro vetor e depois somar cada produto.

Produto Escalar entre 2 Vetores

Definição 2:

$$u = \|\mathbf{w}\| . \|\mathbf{x}\| . \cos(\alpha)$$

Alternativamente, o produto escalar pode ser definido como o produto dos comprimentos dos vetores com o cosseno do menor ângulo entre eles.

OBS: As duas definições são equivalentes! Prove!

Produto Escalar entre 2 Vetores

Exemplo: Calcule o produto escalar dos 2 vetores abaixo usando as duas definições anteriores.

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Definição 1:

$$u = w_0 x_0 + w_1 x_1 = (0)(1) + (1)(1) = 1$$

Definição 2:

$$u = \|\mathbf{w}\| . \|\mathbf{x}\| . \cos(\alpha) = (1) . (\sqrt{2}) . \cos(45) = (1) . (\sqrt{2}) . (\frac{\sqrt{2}}{2}) = 1$$

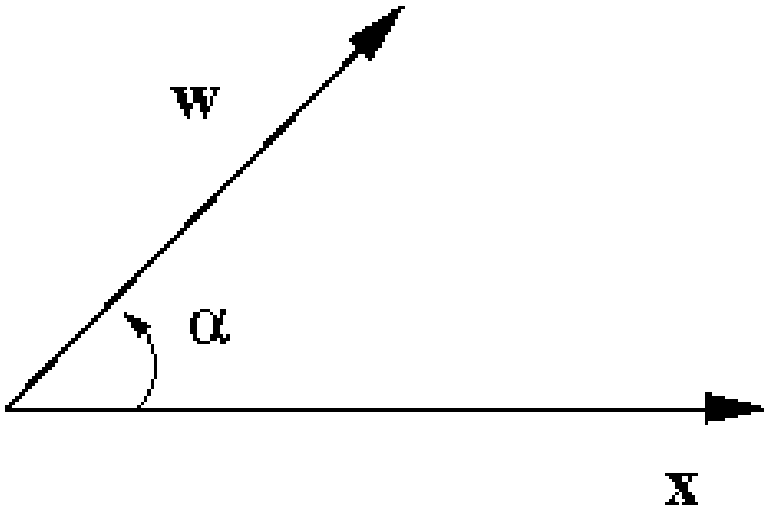
5. Rede Perceptron Simples

Produto Escalar entre 2 Vetores

O produto escalar é uma medida de similaridade entr e vetores.

> "Para vetores de comprimento fixo, quanto menor o ângulo entre eles, maior é o valor resultante do produto escalar".

Exxemplo: $0 \leq \alpha \leq 90^o$



5. Rede Perceptron Simples

Produto Escalar entre 2 Vetores

Vimos que o produto escalar é uma medida de similaridade entr e vetores.

> "Para vetores de comprimento fixo, quanto menor o ângulo entre eles, maior é o valor resultante do produto escalar".

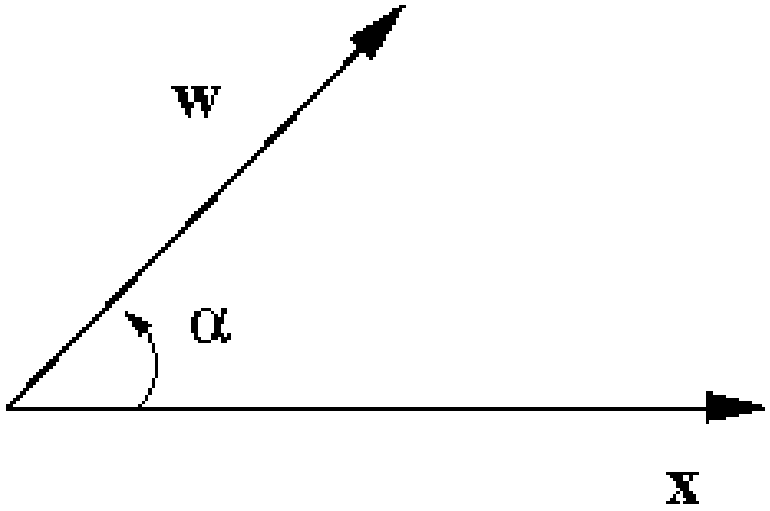
O sinal do produto escalar também é um item importante na análise da orinetação entre os dois vetores.

O sinal depende basicamente do ângulo entre os vetores.

5. Rede Perceptron Simples

Produto Escalar entre 2 Vetores

Caso 1: $0 \leq \alpha < 90^o$

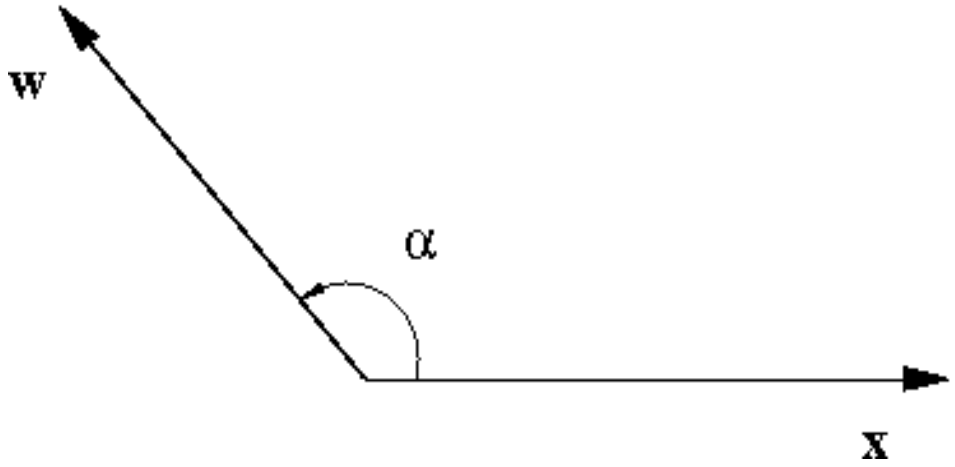


$\cos(\alpha) > 0 \quad \Rightarrow \quad u > 0$ (positivo)

5. Rede Perceptron Simples

Produto Escalar entre 2 Vetores

Caso 2: $90^o < \alpha \leq 180^o$



$\cos(\alpha) < 0 \quad \Rightarrow \quad u < 0$ (negativo)

5. Rede Perceptron Simples

Ativação do Neurônio M-P na Forma Vetorial

$$u = \mathbf{w}^T \mathbf{x} = \mathbf{x}^T \mathbf{w}$$

$$= w_0 x_0 + w_1 x_1 + \ldots + w_p x_p$$

$$= w_1 x_1 + \ldots + w_p x_p - \theta$$

Podemos entender a equação da ativação do neurônio M-P como:

(*i*) uma reta (ou plano) que separa o espaço de entrada em dois semi-planos.

(*ii*) o produto escalar do vetor de entrada (**x**) com o vetor de pesos (**w**).

5. Rede Perceptron Simples

Regra de Aprendizagem do Perceptron Simples

A forma vetorial da ativação (u) nos ajudará no processo de obtenção de uma regra de aprendizagem para o neurônio M-P.

O processo de aprendizagem consiste na modificação dos pesos e do limiar do neurônio M-P até que ele resolva o problema de interesse ou que o período de aprendizagem tenha finalizado.

A regra de aprendizagem é uma função de 2 fatores:

(i) Erro entre a saída desejada (d) e a saída gerada pela rede (y):

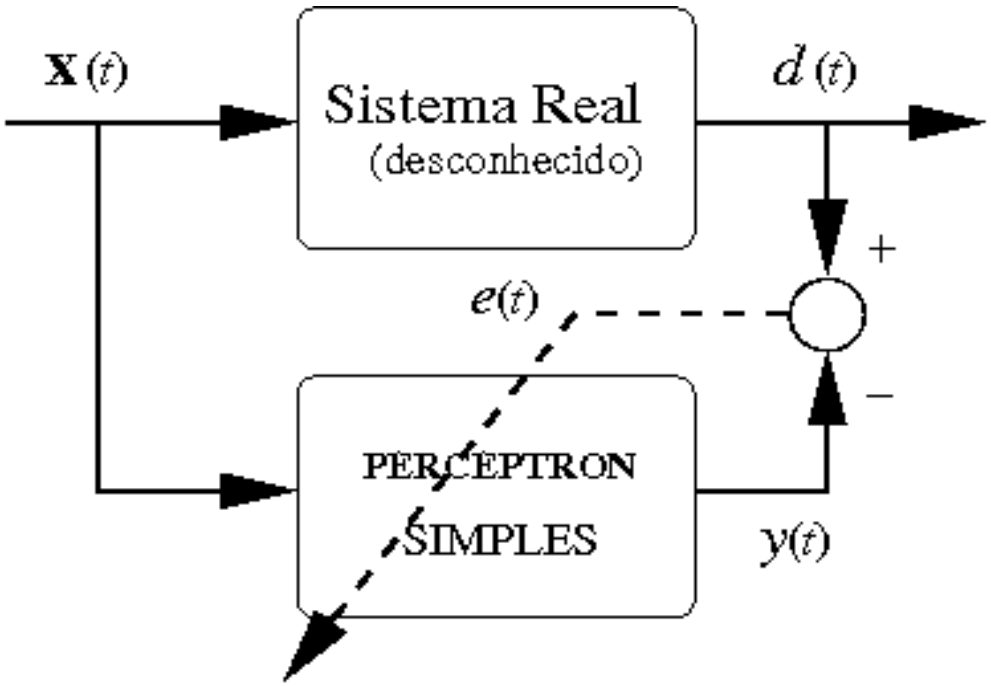
$$e = d - y$$

(ii) Informação fornecida pelo vetor de entrada ($\mathbf{x}$).

5. Rede Perceptron Simples

Regra de Aprendizagem do Perceptron Simples

O processo de aprendizagem, ou seja, de modificação dos parâmetros do neurônio M-P é guiado pelo erro (e) e pelo vetor de entrada ($\mathbf{x}$)!



5. Rede Perceptron Simples

Como projetar então uma regra de aprendizagem?

Uma regra de aprendizagem pode ser projetada com base em

(*i*) Argumentos geométricos ou empíricos.

(*ii*) Critérios de otimização de função-custo.

Em geral, uma regra de aprendizagem tem a seguinte forma:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \Delta\mathbf{w}(t)$$

$\mathbf{w}(t)$ = memória (conhecimento atual).

$\Delta\mathbf{w}(t)$ = incremento na memória (informação adquirida)

$\mathbf{w}(t+1)$ = memória modificada com acrécimo de nova informação.

5. Rede Perceptron Simples

Do exposto em slides anteriores, podemos escrever que:

$$\Delta \mathbf{w}(t) = \mathbf{F}\big(e(t), \mathbf{x}(t)\big)$$

onde t indica o instante de apresentação do vetor de entrada.

Vamos utilizar argumentos geométricos para obter a regra de aprendizagem do neurônio M-P.

Para isso, vamos analisar os possíveis valores que a variável erro (e) pode assumir.

Caso 1: $e = d - y = +1$ (d=+1 e y=0)

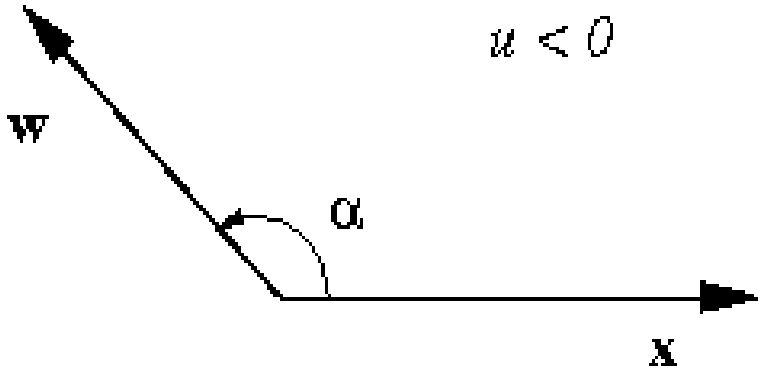
Caso 2: $e = d - y = -1$ (d=0 e y=+1)

Caso 3: $e = d - y = 0$ (d=+1 e y=+1) ou (d=0 e y=0)

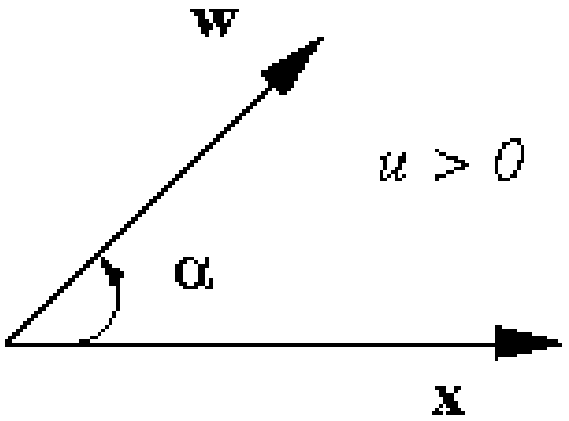
5. Rede Perceptron Simples

Caso 1: $e = d - y = +1$ (d=+1 e y=0)

Situação ocorrida (u<0, y=0):



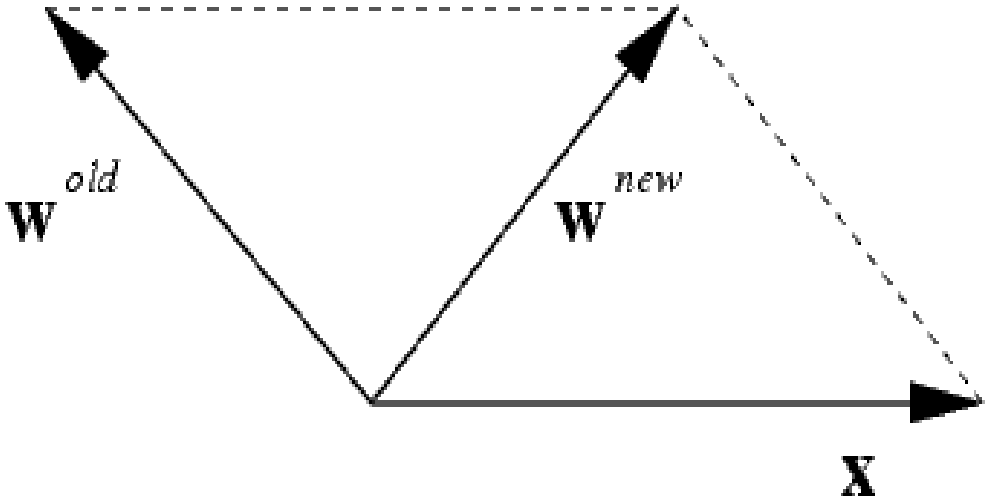
Situação desejada (u>0, y=1):



5. Rede Perceptron Simples

Caso 1 [$e(t) = +1$]: O vetor **w** deve ser modificado para se aproximar de **x**.

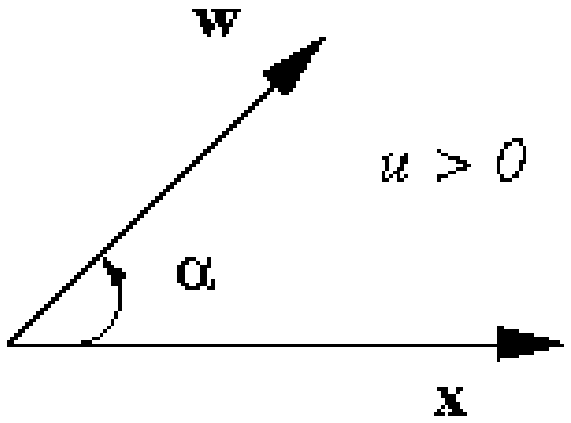
$$\mathbf{w}(t+1) = \mathbf{w}(t) + \mathbf{x}(t)$$



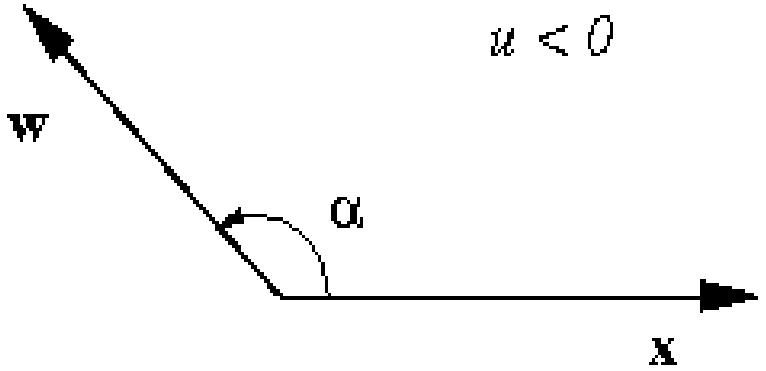
5. Rede Perceptron Simples

Caso 2: $e = d - y = -1$ (d=0 e y=+1)

Situação ocorrida (u>0, y=+1):



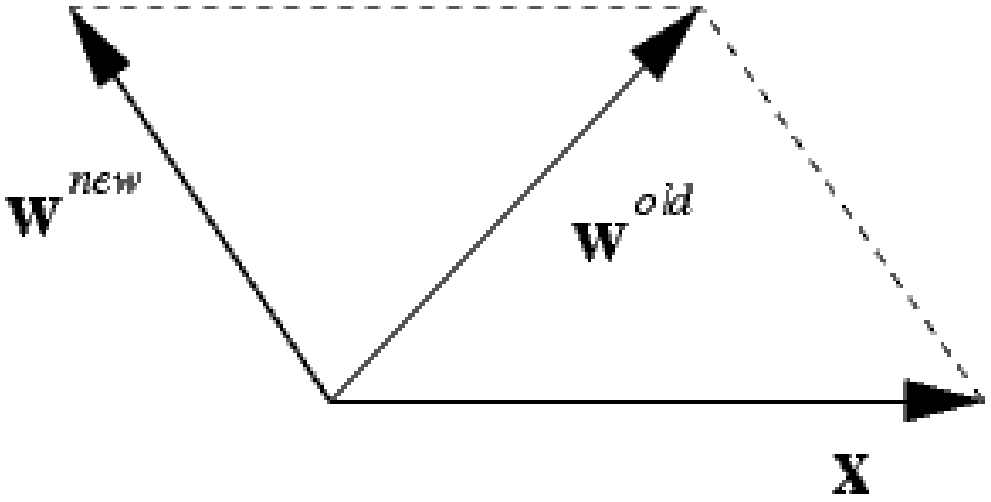
Situação desejada (u<0, y=0):



5. Rede Perceptron Simples

Caso 2 [$e(t) = -1$]: O vetor **w** deve ser modificado para se afastar de **x**.

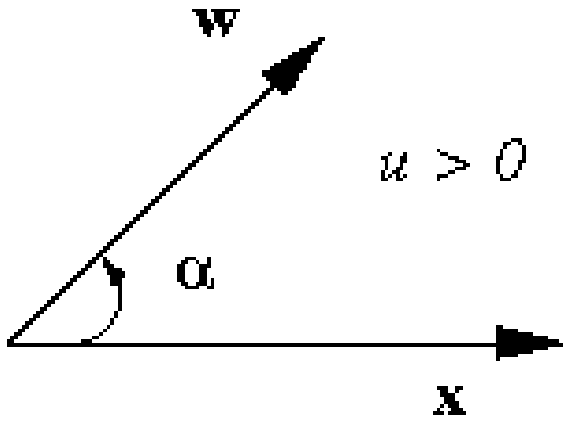
$$\mathbf{w}(t+1) = \mathbf{w}(t) - \mathbf{x}(t)$$



5. Rede Perceptron Simples

Caso 3a: $e = d - y = 0$ (d=+1 e y=+1)

Situação ocorrida = Situação desejada (u>0, y=+1)

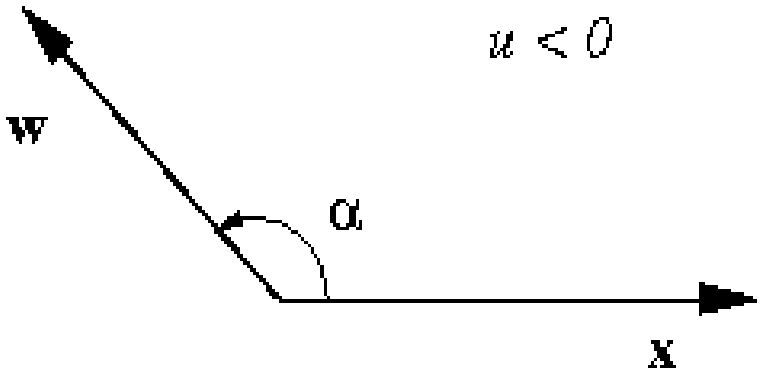


Como houve um acerto, não é preciso modificar o vetor **w**.

5. Rede Perceptron Simples

Caso 3b: $e = d - y = 0$ (d=0 e y=0)

Situação ocorrida = Situação desejada (u<0, y=0)



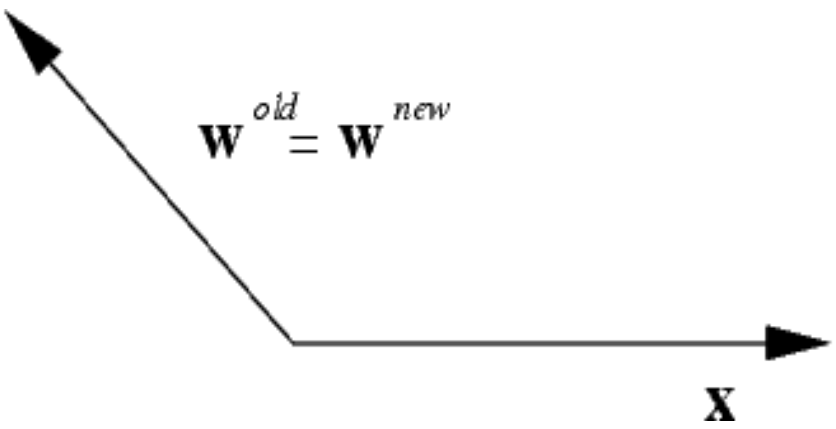
Como houve um acerto, não é preciso modificar o vetor **w**.

5. Rede Perceptron Simples

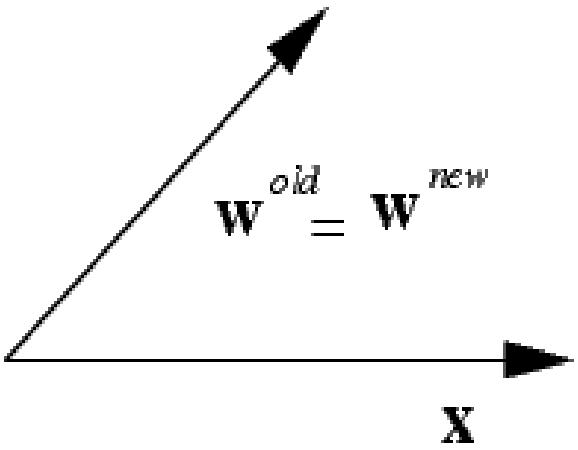
Caso 3 [$e(t) = 0$]: O vetor **w** não deve ser modificado.

$$\mathbf{w}(t+1) = \mathbf{w}(t)$$

Caso 3a



Caso 3b



5. Rede Perceptron Simples

Regra de Aprendizagem do Perceptron

As três equações dos slides anteriores podem ser combinadas em uma única equação que depende do er ro e do vetor de entrada **x**:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + e(t)\mathbf{x}(t)$$

A fim de tornar o processo de ajuste do vetor **w** mais estável, é comum introduzir na equação anterior um fator η, chamado de passo de aprendizagem:

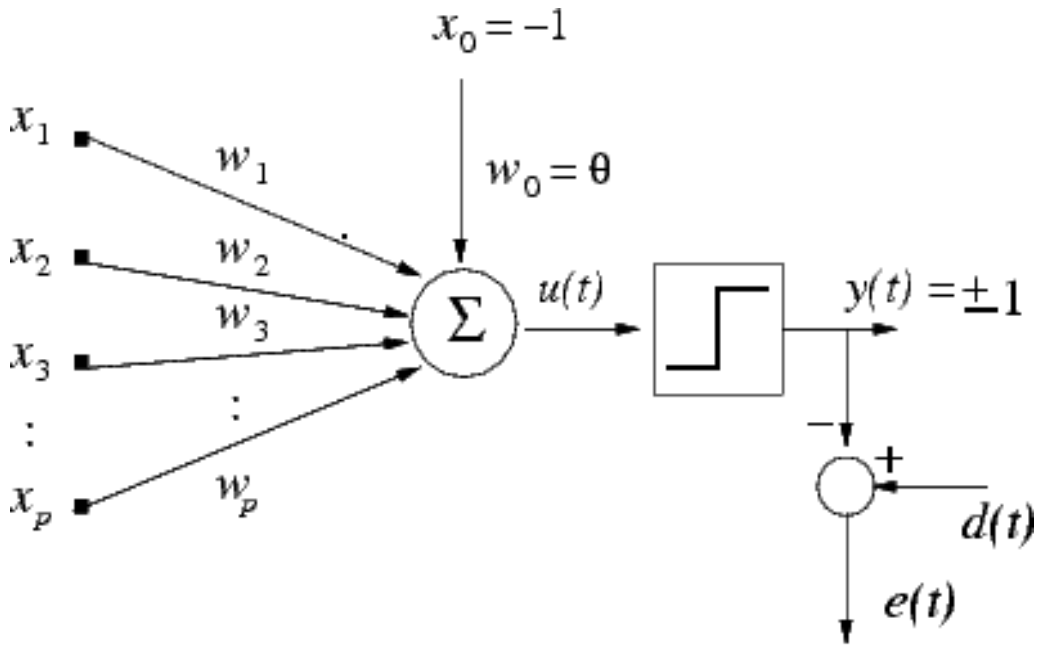
$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta e(t)\mathbf{x}(t)$$

Em que $0 < \eta << 1$.

5. Rede Perceptron Simples

Resumo do Algoritmo do Percepron Simples

Perceptron Simples = Neurônio de M-P + Regra de Aprendizagem



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

5. Rede Perceptron Simples

Resumo do Algoritmo do Percepron Simples

1. Início (t=0)
   - 1.1 – Definir valor de η entre 0 e 1.
   - 1.2 – Iniciar $\mathbf{w}(0)$ com valores nulos ou aleatórios.
2. Funcionamento
   - 2.1 – Selecionar vetor de entrada $\mathbf{x}(t)$.
   - 2.2 – Calcular ativação $u(t)$.
   - 2.3 – Calcular saída $y(t)$.
3. Treinamento
   - 3.1 – Calcular erro: $e(t) = d(t) - y(t)$
   - 3.2 – Ajustar pesos via regra de aprendizagem.
   - 3.3 – Verificar critério de parada.
     - 3.3.1 – Se atendido, finalizar treinamento.
     - 3.3.2 – Caso contrário, fazer t=t+1 e ir para Passo 2.

5. Rede Perceptron Simples

Exemplo Passo-a-Passo: Aprendendo a Porta Lógica OR.

Porta OR

x_1	x_2	d
0	0	0
0	1	1
1	0	1
1	1	1

t =0: Iniciar com zeros os pesos e o limiar.

$$w_1(0)=w_2(0)=\theta(0)=0$$

Logo:

$$\mathbf{w}(0)=\begin{bmatrix}\theta(0)\\ w_1(0)\\ w_2(0)\end{bmatrix}=\begin{bmatrix}0\\ 0\\ 0\end{bmatrix} \qquad \mathbf{x}(t)=\begin{bmatrix}-1\\ x_1(t)\\ x_2(t)\end{bmatrix}$$

Passo de aprendizagem escolhido: $\eta = 0{,}5$;

5. Rede Perceptron Simples

t=1: Calcular saída para $\mathbf{w}(1) = [0 \;\; 0 \;\; 0]$ e $\mathbf{x}(1) = [-1 \;\; 0 \;\; 0]$.

$$u(1) = (0)(-1) + (0)(0) + (0)(0) = 0 \Rightarrow y(1) = 0,\ e(1) = 0.$$

$$\mathbf{w}(2) = \mathbf{w}(1) + \eta\, e(1)\, \mathbf{x}(1) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + (0{,}5)(0) \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \mathbf{w}(1)$$

t=2: Calcular saída para $\mathbf{w}(2) = [0 \;\; 0 \;\; 0]$ e $\mathbf{x}(2) = [-1 \;\; 0 \;\; 1]$.

$$u(2) = (0)(-1) + (0)(0) + (0)(1) = 0 \Rightarrow y(2) = 0,\ e(2) = 1$$

$$\mathbf{w}(3) = \mathbf{w}(2) + \eta\, e(2)\, \mathbf{x}(2) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + (0{,}5)(1) \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -0{,}5 \\ 0 \\ 0{,}5 \end{bmatrix}$$

5. Rede Perceptron Simples

t=3: Calcular saída para $\mathbf{w}(3) = [-0,5 \;\; 0 \;\; 0,5]$ e $\mathbf{x}(3) = [-1 \;\; 1 \;\; 0]$.

$u(3)=(-0,5)(-1)+(0)(1) + (0,5)(0) = 0,5 \Rightarrow y\,(3)= 1$, $e(1)=0$.

$\mathbf{w}(4)=\mathbf{w}(3)$

t=4: Calcular saída para $\mathbf{w}(4) = [-0,5 \;\; 0 \;\; 0,5]$ e $\mathbf{x}(4) = [-1 \;\; 1 \;\; 1]$.

$u(4) = (-0,5)(-1) + (0)(1) + (0,5)(1) = 1 \Rightarrow y(4) = 1$, $e(4)=0$.

$\mathbf{w}(5)=\mathbf{w}(4)$

t=5: Calcular saída para $\mathbf{w}(5) = [-0,5 \;\; 0 \;\; 0,5]$ e $\mathbf{x}(5) = [-1 \;\; 0 \;\; 0]$.

$u(5) = (-0,5)(-1) + (0)(0) + (0,5)(0) = 0,5 \Rightarrow y(5) = 1$,

$e(5)=-1$.

$$\mathbf{w}(6)=\mathbf{w}(5)+\eta\, e(5)\mathbf{x}(5)=\begin{bmatrix}-0,5\\0\\0,5\end{bmatrix}+(0,5)(-1)\begin{bmatrix}-1\\0\\0\end{bmatrix}=\begin{bmatrix}0\\0\\0,5\end{bmatrix}$$

5. Rede Perceptron Simples

t=6: Calcular saída para $\mathbf{w}(6) = [0 \;\; 0 \;\; 0{,}5]$ e $\mathbf{x}(6) = [-1 \;\; 0 \;\; 1]$.

$u(6)=(0)(-1)+(0)(0)+(0{,}5)(1) = 0{,}5 \Rightarrow y(6)= 1,\ e(6)=0.$

$\mathbf{w}(7)=\mathbf{w}(6)$

t=7: Calcular saída para $\mathbf{w}(7) = [0 \;\; 0 \;\; 0{,}5]$ e $\mathbf{x}(7) = [-1 \;\; 1 \;\; 0]$.

$u(7) = (0)(-1) + (0)(1) + (0{,}5)(0) = 0 \Rightarrow y(7) = 0,\ e(7)=1.$

$$\mathbf{w}(8)=\mathbf{w}(7)+\eta\, e(7)\,\mathbf{x}(7)=\begin{bmatrix}0\\0\\0{,}5\end{bmatrix}+(0{,}5)(1)\begin{bmatrix}-1\\1\\0\end{bmatrix}=\begin{bmatrix}-0{,}5\\0{,}5\\0{,}5\end{bmatrix}$$

t=8: Calcular saída para $\mathbf{w}(8) = [-0{,}5 \;\; 0{,}5 \;\; 0{,}5]$ e $\mathbf{x}(8) = [-1 \;\; 1 \;\; 1]$.

$u(8) = (-0{,}5)(-1)+(0{,}5)(1)+(0{,}5)(1) = 0{,}5 \Rightarrow y(8) = 1,\ e(8)=0.$

$\mathbf{w}(9)=\mathbf{w}(8)$

5. Rede Perceptron Simples

t=9: Calcular saída para $\mathbf{w}(9) = [-0{,}5\ \ 0{,}5\ \ 0{,}5]$ e $\mathbf{x}(9) = [-1\ \ 0\ \ 0]$.

$u(9) = (-0{,}5)(-1) + (0{,}5)(0) + (0{,}5)(0) = 0{,}5 \Rightarrow y(9) = 1,\ e(9) = -1.$

$$\mathbf{w}(10) = \mathbf{w}(9) + \eta\, e(9)\, \mathbf{x}(9) = \begin{bmatrix} -0{,}5 \\ 0{,}5 \\ 0{,}5 \end{bmatrix} + (0{,}5)(-1) \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0{,}5 \\ 0{,}5 \end{bmatrix}$$

t=10: Calcular saída para $\mathbf{w}(10) = [0\ \ 0{,}5\ \ 0{,}5]$ e $\mathbf{x}(10) = [-1\ \ 0\ \ 1]$.

$u(7) = (0)(-1) + (0{,}5)(0) + (0{,}5)(1) = 0{,}5 \Rightarrow y(10) = 1,\ e(10) = 0.$

$\mathbf{w}(11) = \mathbf{w}(10)$

t=11: Calcular saída para $\mathbf{w}(11) = [0\ \ 0{,}5\ \ 0{,}5]$ e $\mathbf{x}(11) = [-1\ \ 1\ \ 0]$.

$u(11) = (0)(-1) + (0{,}5)(1) + (0{,}5)(0) = 0{,}5 \Rightarrow y(11) = 1,\ e(11) = 0.$

$\mathbf{w}(12) = \mathbf{w}(11)$

5. Rede Perceptron Simples

t=12: Calcular saída para $\mathbf{w}(12) = [0 \;\; 0{,}5 \;\; 0{,}5]$ e $\mathbf{x}(12) = [-1 \;\; 1 \;\; 1]$.

$u(12)=(0)(-1)+(0{,}5)(1) + (0{,}5)(1) = 1 \Rightarrow y\,(12)= 1, e(12)=0.$

$\mathbf{w}(13)=\mathbf{w}(12)$

t=13: Calcular saída para $\mathbf{w}(13) = [0 \;\; 0{,}5 \;\; 0{,}5]$ e $\mathbf{x}(13) =[-1 \;\; 0 \;\; 0]$.

$u(13) = (0)(-1) + (0{,}5)(0) + (0{,}5)(0) = 0 \Rightarrow y(13) = 0, e(13)=0.$

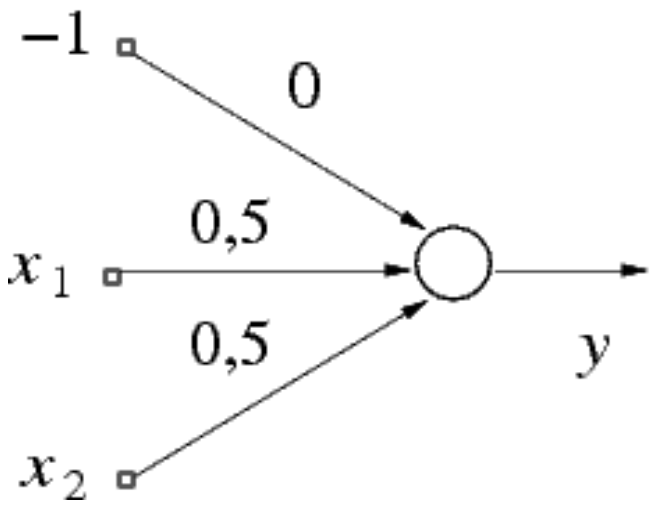
$\mathbf{w}(14)=\mathbf{w}(13)$

FIM do treinamento!

5. Rede Perceptron Simples

Solução Encontrada:

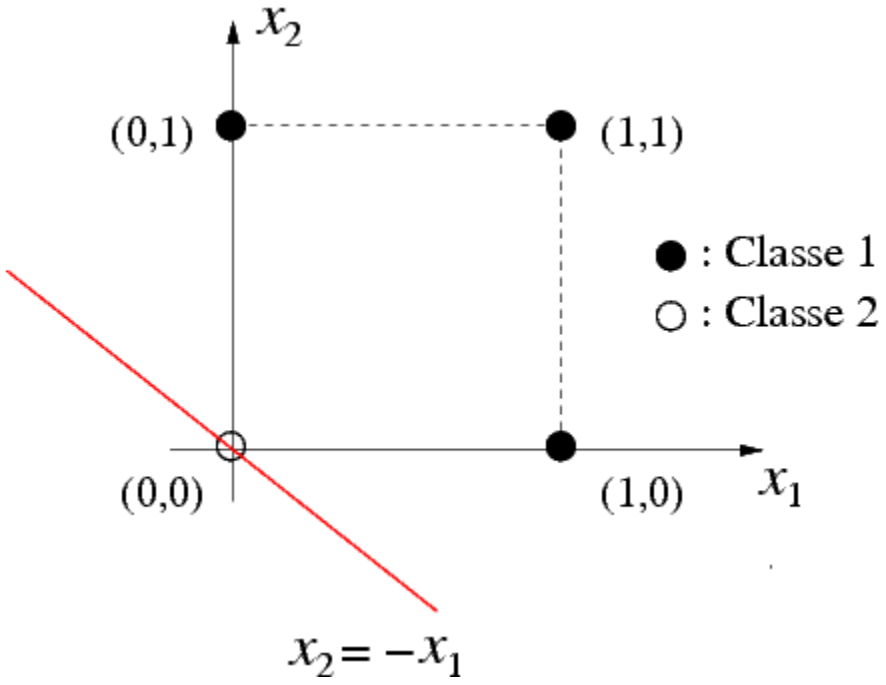
$$\mathbf{w} = \begin{bmatrix} \theta \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0{,}5 \\ 0{,}5 \end{bmatrix}$$



5. Rede Perceptron Simples

Solução Encontrada:

$$\mathbf{w} = \begin{bmatrix} \theta \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0{,}5 \\ 0{,}5 \end{bmatrix}$$

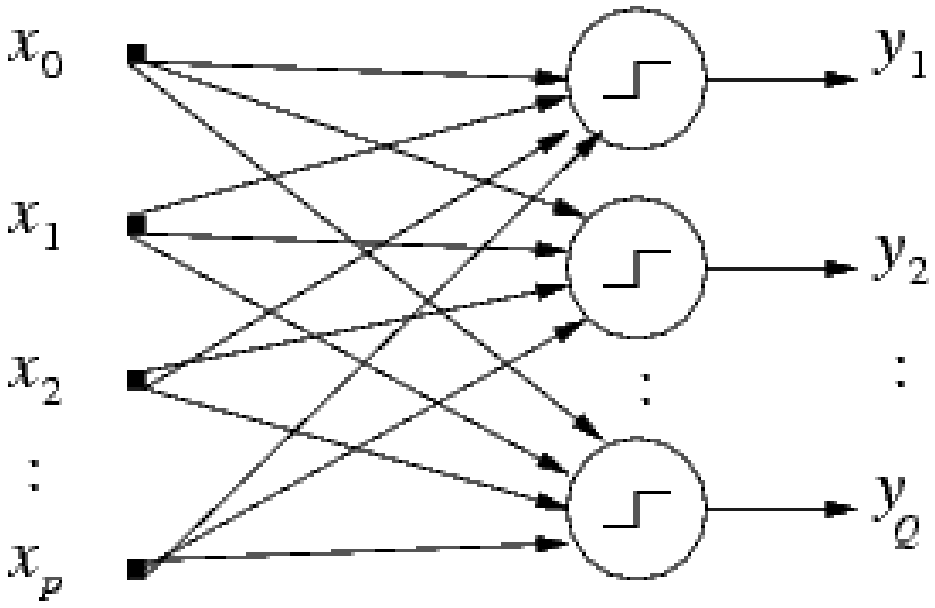


Note que esta não é a melhor das soluções, porque a reta passa bem em cima do ponto (0,0). Se os pesos tivessem sido iniciados aleatoriamente, dificilmente uma situação como essa ocorreria.

5. Rede Perceptron Simples

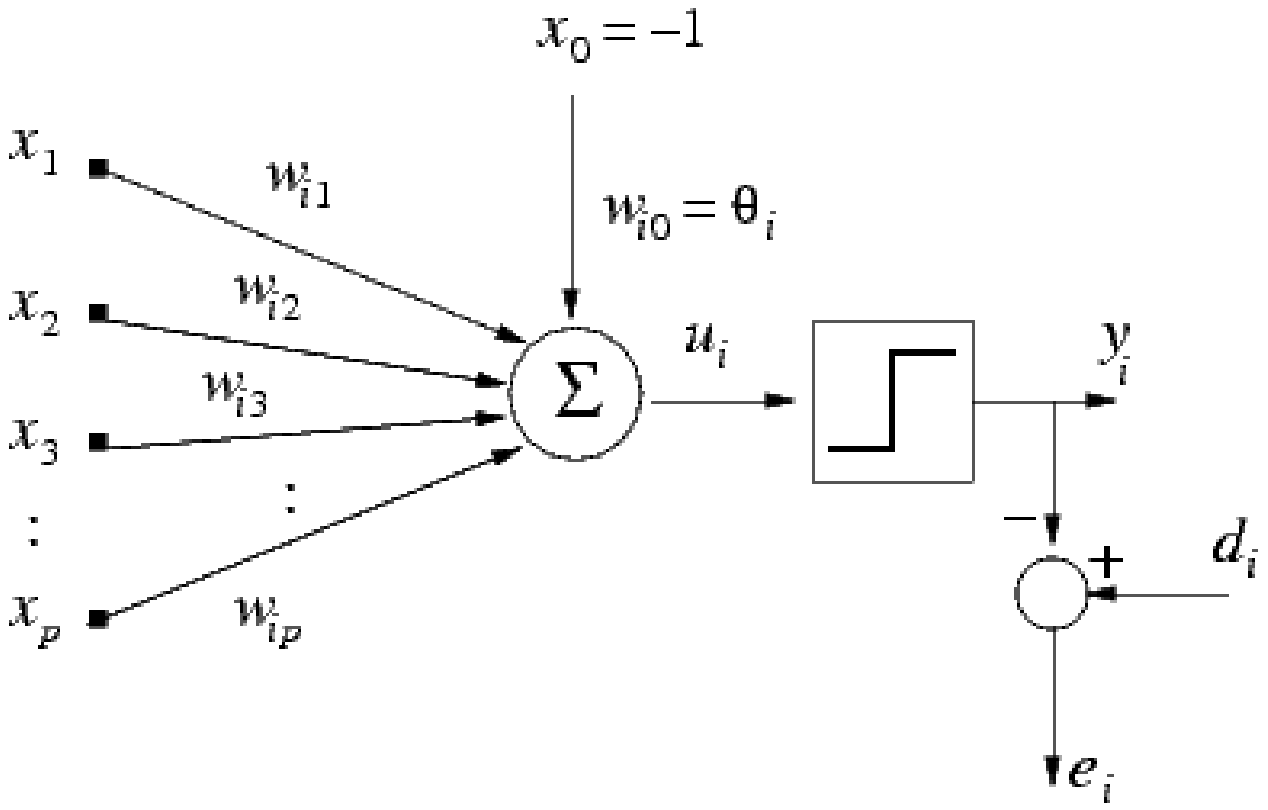
Um único neurônio M-P categoriza apenas duas classes de dados.

Em problemas com múltiplas classes, deve-se utilizar vários neurônios em paralelo.



5. Rede Perceptron Simples

O *i*-ésimo neurônio da rede PS é representado na figura abaixo.



5. Rede Perceptron Simples

O funcionamento de cada neurônio individualmente é o mesmo.

Assim, a ativação do *i*-ésimo neurônio da rede PS é dada por:

$$u_i = \mathbf{w}_i^T \mathbf{x} = w_{i1} x_1 + w_{i2} x_2 + \ldots + w_{ip} x_p$$

A saída do *i*-ésimo neurônio é dada por:

$$y_i = \text{sinal}(u_i) = \text{sinal}(\mathbf{w}_i^T \mathbf{x})$$

O erro do *i*-ésimo neurônio é dado por: $e_i = d_i - y_i$

onde d_i é a saída desejada do *i*-ésimo neurônio.

$i = 1, \ldots, Q$ ($Q \geq 1$ é o número de neurônios de saída).

5. Rede Perceptron Simples

Como cada neurônio tem seu próprio vetor de pesos $\mathbf{w}_i$, $i = 1, 2, ..., Q$, então teremos agora Q regras de aprendizagem!

Ou seja, uma regra de aprendizagem para cada vetor $\mathbf{w}_i$.

Assim, a regra de aprendizagem do i-ésimo neurônio é dada por:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \eta\, e_i(t)\, \mathbf{x}(t)$$

Em que $0 < \eta << 1$ e $i=1, 2, ..., Q$.

5. Rede Perceptron Simples

Resumo da Rede Percepron Simples (*Q* neurônios)

1. Início (t=0)
   - 1.1 – Definir valor de η entre 0 e 1.
   - 1.2 – Iniciar $\mathbf{w}_i(0)$ com valores aleatórios.
2. Funcionamento
   - 2.1 – Selecionar o vetor de entrada $\mathbf{x}(t)$.
   - 2.2 – Calcular as Q ativações $u_i(t)$.
   - 2.3 – Calcular as Q saídas $y_i(t)$.
3. Treinamento
   - 3.1 – Calcular os Q erros: $e_i(t) = d_i(t) - y_i(t)$
   - 3.2 – Ajustar os Q vetores de pesos $\mathbf{w}_i(t)$.
   - 3.3 – Verificar critério de parada.
     - 3.3.1 – Se atendido, finalizar treinamento.
     - 3.3.2 – Caso contrário, fazer t=t+1 e ir para Passo 2.

5. Rede Perceptron Simples

Para o PS, existem basicamente 2 métodos para especificar Q.

Método 1: Codificação binária simples.

Se tenho C classes, então Q é o maior inteiro igual a ou menor que $\sqrt{C}$.

Exemplo: Se $C = 6$ classes, então $Q > 2{,}45 = 3$.

Os vetores de saídas desejadas são construídos do seguinte modo:

Classe 1: $\mathbf{d} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ Classe 2: $\mathbf{d} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ Classe 3: $\mathbf{d} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$

E assim por diante até a Classe 6: $\mathbf{d} = [1 \;\; 1 \;\; 0]^T$.

5. Rede Perceptron Simples

Método 2: Codificação 1-*out-of-Q*.

Se tenho C classes, então $Q = C$.

Exemplo: Se $C = 4$ classes, então $Q = 4$.

Neste método apenas uma das componentes do vetor de saídas desejadas tem valor igual a 1, i.e. Os vetores **d** são ortogonais.

Classe 1: $\mathbf{d} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ Classe 2: $\mathbf{d} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ Classe 3: $\mathbf{d} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$

E assim por diante até a Classe 4: $\mathbf{d} = [0 \;\; 0 \;\; 0 \;\; 1]^T$.

5. Rede Perceptron Simples

Dicas para Projetar uma Rede PS

(1) Usar uma taxa de aprendizagem pequena (e.g. η= 0,1 ou 0,01).

(2) Usar valores de saída $y_i \in \{-1,+1\}$, em vez de $y_i \in \{0,+1\}$.

(3) Mudar a ordem de apresentação dos vetores de treinamento a cada época de treinamento, tornando-a aleatória.

(4) Usar o método dois para determinar o número de neurônios *(Q)* e a representação dos vetores de saídas desejadas **(d)**.

Dicas para Projetar uma Rede PS

(5) Normalizar os vetores de entrada se as variáveis apresentarem ordens de grandeza muito díspares.

Recomenda-se deixar toda variável com valores

dentro da faixa [0,+1]:

$$x_j^{norm} = \frac{x_j - x_j^{min}}{x_j^{max} - x_j^{min}}$$

ou dentro da faixa [-1,+1]:

$$x_j^{norm} = 2 . \left(\frac{x_j - x_j^{min}}{x_j^{max} - x_j^{min}} \right) - 1$$

5. Rede Perceptron Simples

Exemplo Prático

(1) Problema de auxílio ao diagnóstico em dermatologia.

(2) Número de classes igual a $C = 6$.

Classe	Patologia	No. de casos
1	psoriasis	112
2	seboreic dermatitis	61
3	lichen planus	72
4	pityriasis rosea	49
5	cronic dermatitis	52
6	pityriasis rubra pilaris	20

5. Rede Perceptron Simples

(3) Número total de casos clínicos $N = 366$ (8 com informação faltante).

(4) Porcentagem de casos usados para treinamento = 80%.

(5) Representação da saída via Método 2, logo $Q = 6$.

(6) Doadores dos Dados

-- 1. Nilsel Ilter, M.D., Ph.D.,
Gazi University,
School of Medicine
06510 Ankara, Turkey

-- 2. H. Altay Guvenir, PhD.,
Bilkent University,
Dept. Computer Engineering
06533 Ankara, Turkey

5. Rede Perceptron Simples

(7) No. de variáveis de entrada (p=34)

Atributos Clínicos (assumem valores 0, 1, 2, 3, salvo indicação contrária)

1: erythema
2: scaling
3: definite borders
4: itching
5: koebner phenomenon
6: polygonal papules
7: follicular papules
8: oral mucosal involvement
9: knee and elbow involvement
10: scalp involvement
11: family history, (0 or 1)
34: Age (linear)

5. Rede Perceptron Simples

Atributos Histopatológicos 1 (assumem valores 0, 1, 2, 3, 4)

12: melanin incontinence
13: eosinophils in the infiltrate
14: PNL infiltrate
15: fibrosis of the papillary dermis
16: exocytosis
17: acanthosis
18: hyperkeratosis
19: parakeratosis
20: clubbing of the rete ridges
21: elongation of the rete ridges

Ementa

1. Funcionalidades do neurônio biológico
2. Neurônio artificial de McCulloch-Pitts (MP)
3. Análise geométrica do neurônio MP
4. Portas lógicas AND, OR e NOT
5. Rede Perceptron Simples (PS) e aplicações
6. **Problemas não-linearmente separáveis (porta XOR)**
7. **Implementação da porta XOR via redes multicamadas**
8. Rede Perceptron Multicamadas (MLP)
9. Algoritmo de retropropagação do erro (*error backpropagation*)
10. Dicas de treinamento, teste e validação da rede MLP

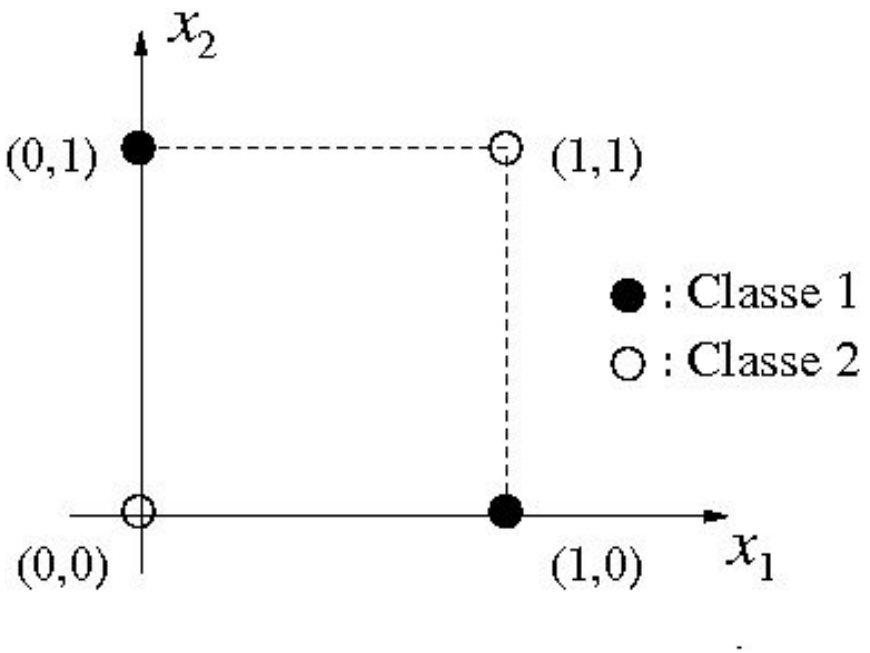
6. Problemas Não-Linearmente Separáveis (Porta XOR)

Exemplo: É possível implementar a porta XOR com 1 neurônio?

Representação do Problema (Função XOR)

Porta XOR

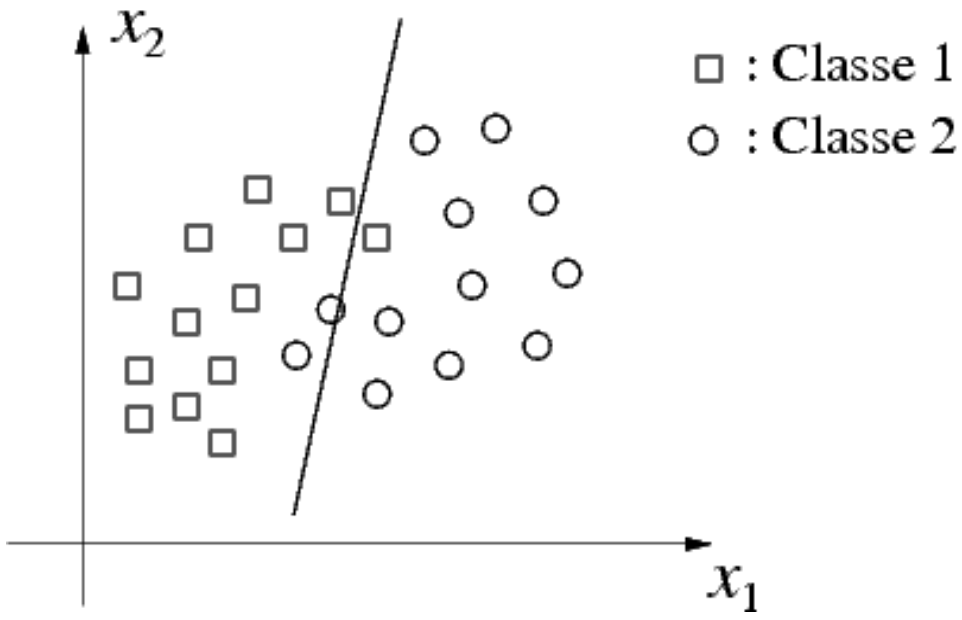
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



6. Problemas Não-Linearmente Separáveis (Porta XOR)

***Exemplo* (cont.)** : Não, porque a função lógica XOR é

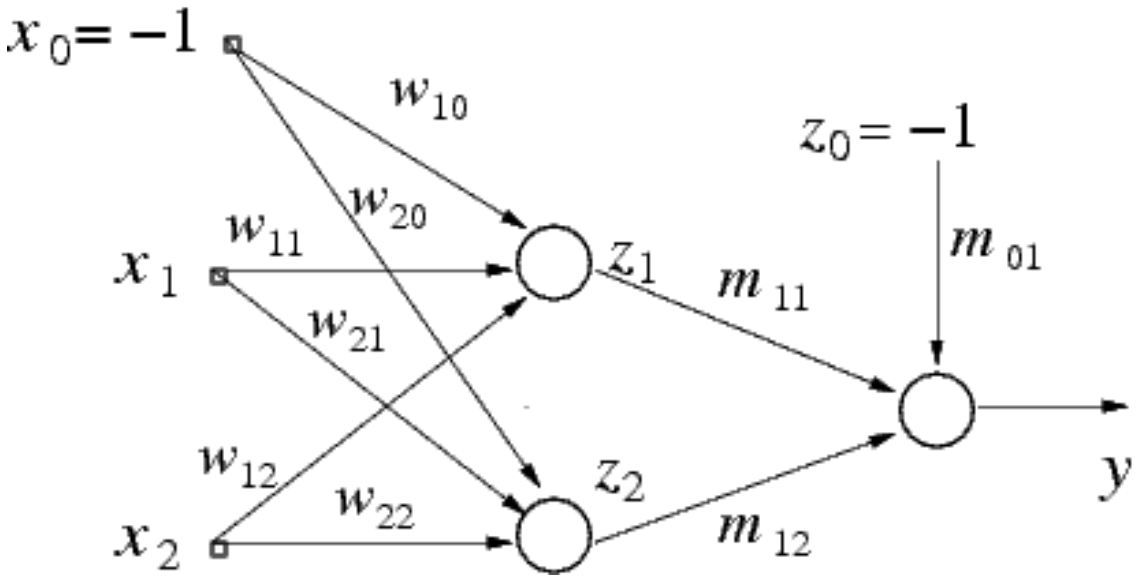
não-linearmente separável.



Ou seja, não é possível separar as classes por uma única reta.

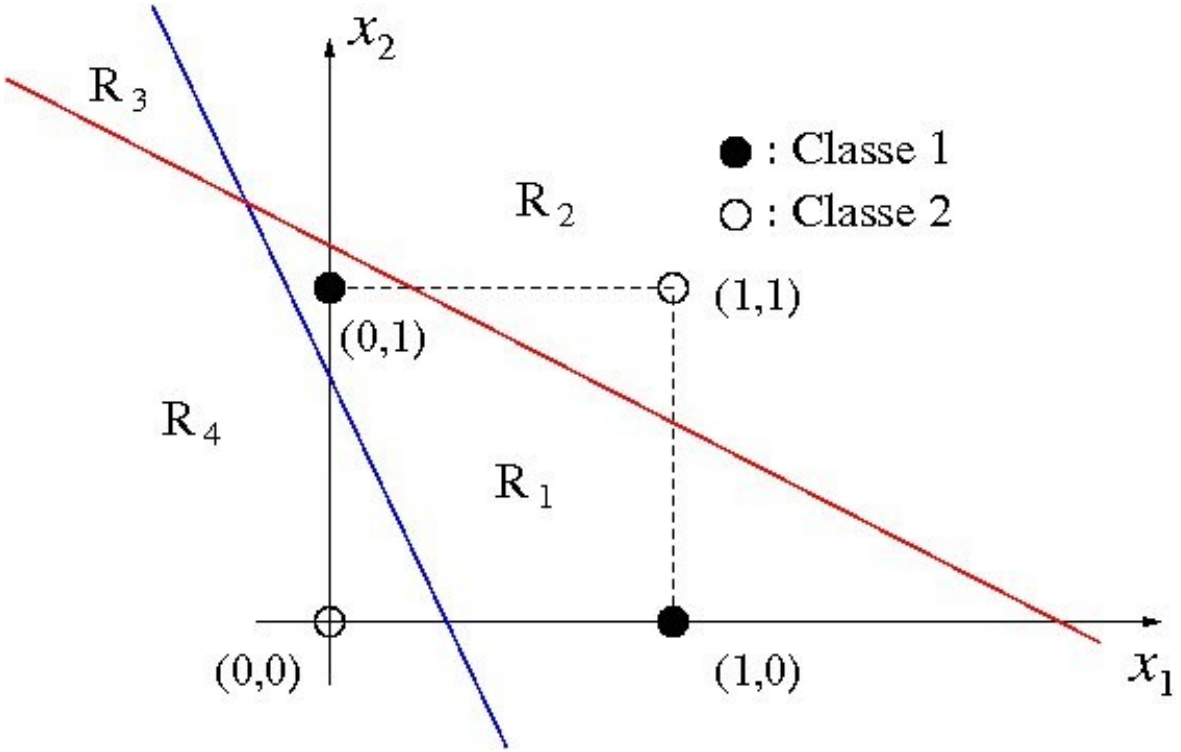
6. Problemas Não-Linearmente Separáveis (Porta XOR)

***Exemplo* (cont.)** : São necessários pelo menos TRÊS neurônios!!



6. Problemas Não-Linearmente Separáveis (Porta XOR)

***Exemplo* (cont.)** : Dois neurônios são necessários para separar o espaço (x_1, x_2) em 4 regiões (R_1, R_2, R_3, R_4).



6. Problemas Não-Linearmente Separáveis (Porta XOR)

***Exemplo* (cont.)** : Note que a reta em vermelho corresponde a um neurônio que implementa a porta AND, enquanto a reta em azul corresponde a um neurônio que implementa a porta OR.

Sejam z_1 e z_2, as saídas dos neurônios responsáveis pelas retas em vermelho e azul, respectivamente. Assim, temos que:

Em R_1, $z_1 = 0$ e $z_2 = 1$. Em R_2, $z_1 = 1$ e $z_2 = 1$.

Em R_3, $z_1 = 1$ e $z_2 = 0$. Em R_4, $z_1 = 0$ e $z_2 = 0$.

6. Problemas Não-Linearmente Separáveis (Porta XOR)

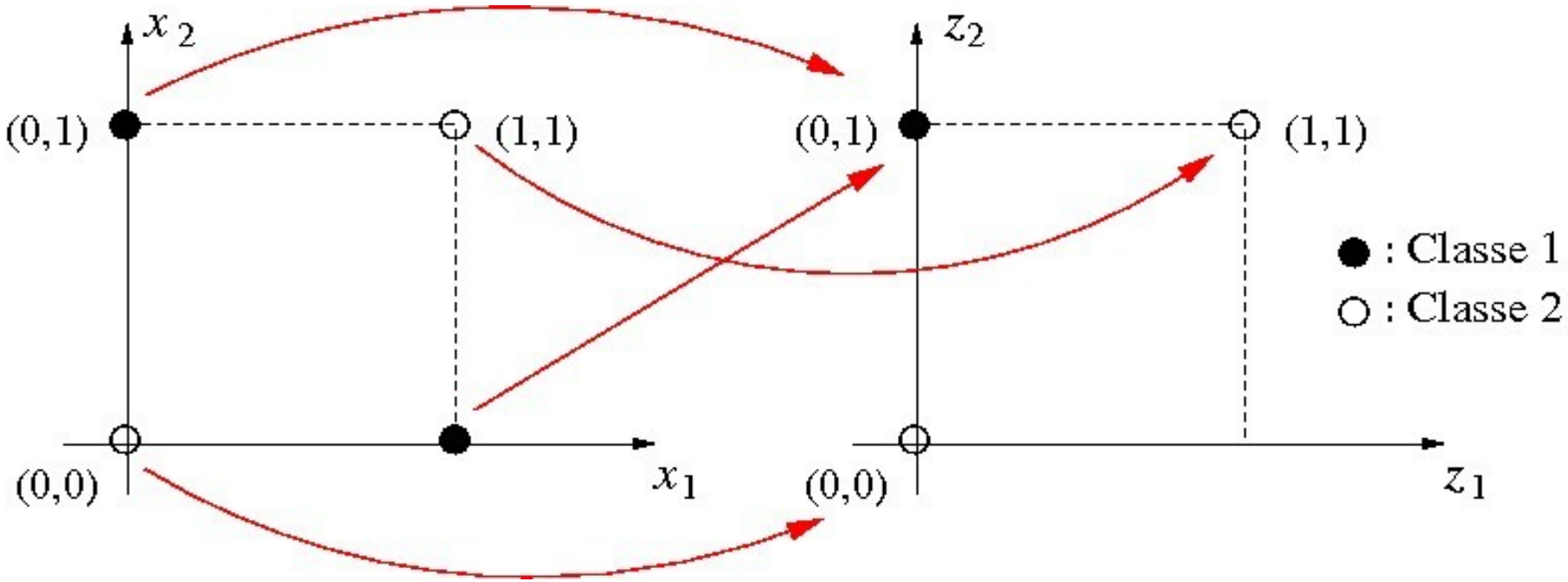
***Exemplo* (cont.)** : Precisa-se ainda de um terceiro neurônio para combinar os valores de z_1 e z_2, a fim de gerar o valor de y correto.

Quando um ponto da função que cair em qualquer uma das regiões R_2, R_3 e R_4, deve gerar uma saída igual a $y = 0$.

Quando um ponto da função que cair em na região R_1, o terceiro neurônio deve gerar uma saída igual a $y = 1$.

6. Problemas Não-Linearmente Separáveis (Porta XOR)

***Exemplo* (cont.)** : Representação da função XOR no espaço (z_1, z_2).



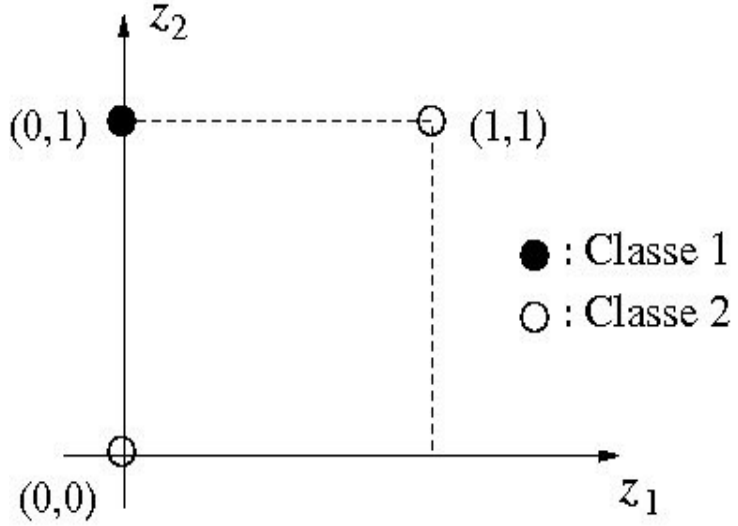
No espaço (z_1, z_2) a função XOR passa a ser linearmente separável, pois o ponto $(x_1, x_2) = (1,0)$ é mapeado no ponto $(z_1, z_2) = (0,1)$!

6. Problemas Não-Linearmente Separáveis (Porta XOR)

***Exemplo* (cont.)** : Assim, devemos projetar um neurônio que implemente a seguinte função lógica no espaço (z_1, z_2).

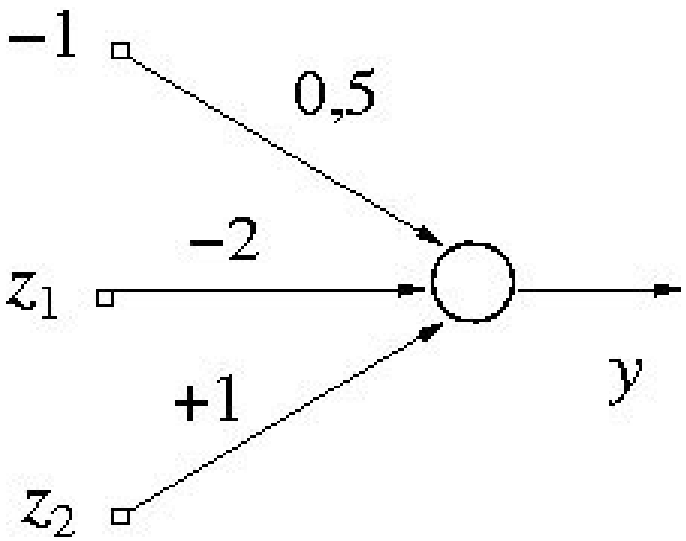
Porta logica

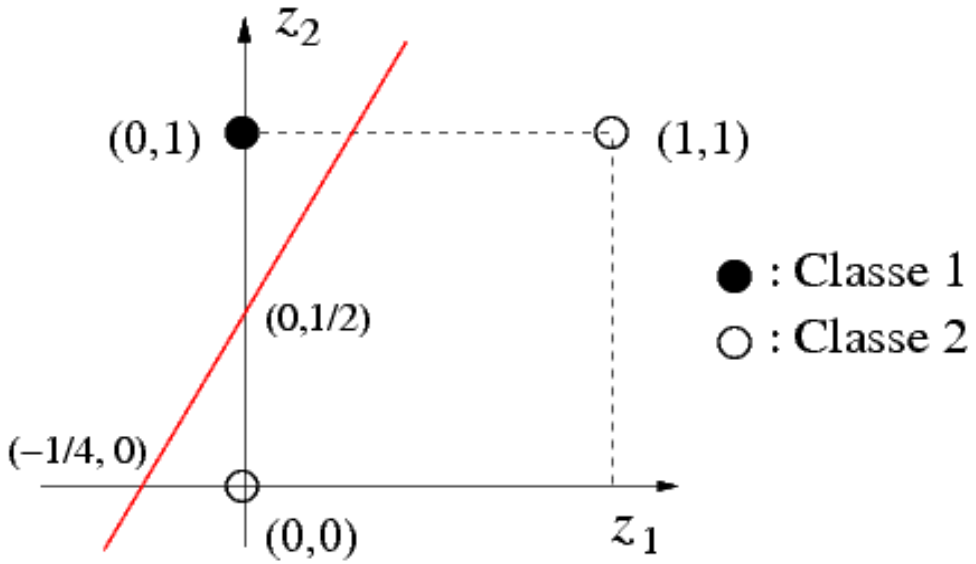
z_1	z_2	y
0	0	0
0	1	1
1	1	0



6. Problemas Não-Linearmente Separáveis (Porta XOR)

***Exemplo* (cont.)**: O neurônio de saída tem a seguinte configuração.

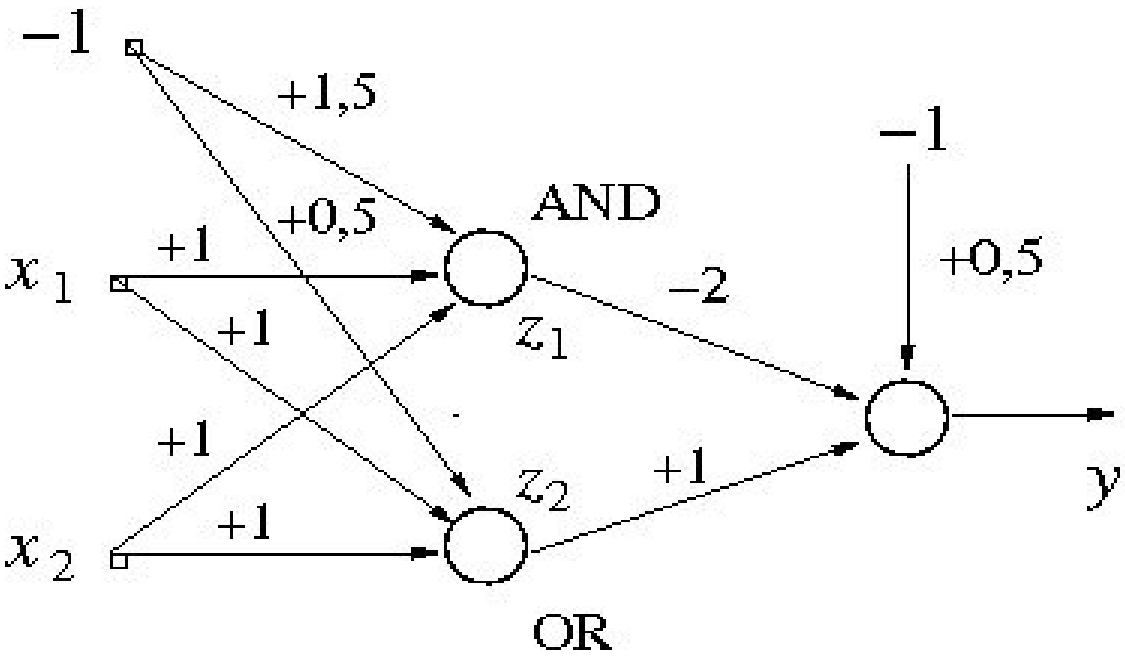




Equação da reta: $z_2 = 2\,z_1 + 0{,}5$

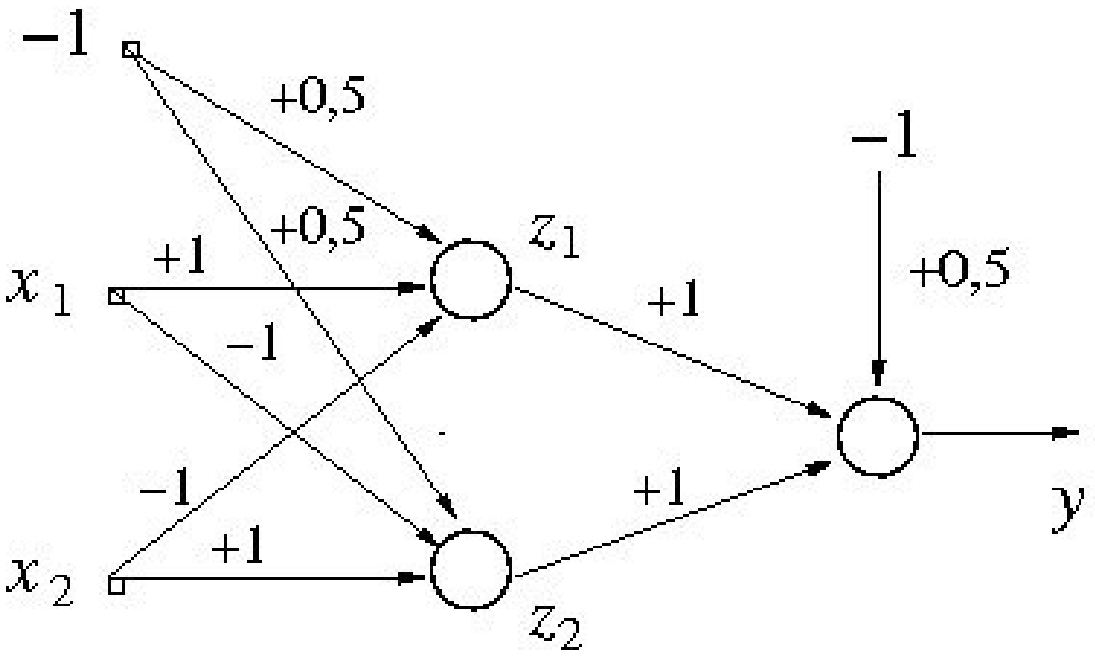
7. Implementação da Porta XOR

***Exemplo* (cont.)** : Colocando os dois primeiros neurônios em uma camada e o terceiro neurônio na camada seguinte (subseqüente), chega-se à seguinte rede multicamadas que implementa a porta XOR.



7. Implementação da Porta XOR

***Exemplo* (cont.)** : Uma outra possível rede multicamadas que também implementa a porta lógica XOR é dada abaixo.



Ementa

1. Funcionalidades do neurônio biológico
2. Neurônio artificial de McCulloch-Pitts (MP)
3. Análise geométrica do neurônio MP
4. Portas lógicas AND, OR e NOT
5. Rede Perceptron Simples (PS) e aplicações
6. Problemas não-linearmente separáveis (porta XOR)
7. Implementação da porta XOR via redes multicamadas
8. **Rede Perceptron Multicamadas (MLP)**
9. Algoritmo de retropropagação do erro (*error backpropagation*)
10. Dicas de treinamento, teste e validação da rede MLP

8. Rede Perceptron Multicamadas

A rede neural conhecida como Perceptron Multicamadas (*Multilayer Perceptron* – MLP) contém os seguintes elementos:

(*i*) **Unidades de entrada**: responsáveis pela simples passagem dos valores de entrada para os neurônios das camadas seguintes.

(*ii*) **Camada(s) oculta(s)**: contém neurônios responsáveis pelo processamento não-linear da informação de entrada, de modo a facilitar a resolução do problema para os neurônios da camada de saída .

(*iii*) **Camada de saída**: contém neurônios responsáveis pela geração da saída da rede neural, após as entradas terem sido devidamente processadas pelos neurônios ocultos.

8. Rede Perceptron Multicamadas

Uma Rede MLP com 1 camada oculta é representada por:

$$\text{MLP}(p, q_1, m)$$

Onde: p é o número de variáveis de entrada
q_1 é o número de neurônios ocultos
m é o número de neurônios de saída.

Logo, o número total de parâmetros (Z) de uma rede MLP de uma camada oculta é dado por:

$$Z = (p+1)q_1 + (q_1+1)m$$

8. Rede Perceptron Multicamadas

Uma Rede MLP com 1 camada oculta é representada por:

$$\text{MLP}(p, q_1, q_2, m)$$

Onde:
- p é o número de variáveis de entrada
- q_1 é o número de neurônios da 1a. camada oculta
- q_2 é o número de neurônios da 2a. camada oculta
- m é o número de neurônios de saída.

Logo, o número total de parâmetros (Z) de uma rede MLP de duas camadas ocultas é dado por:

$$Z = (p+1)q_1 + (q_1+1)q_2 + (q_2+1)m$$

8. Rede Perceptron Multicamadas

Uma rede MLP com 4 variáveis de entrada (p=4), 10 neurônios ocultos (q_1=10) e 2 neurônios de saída (m=2) é representada como MLP(4,10,2).

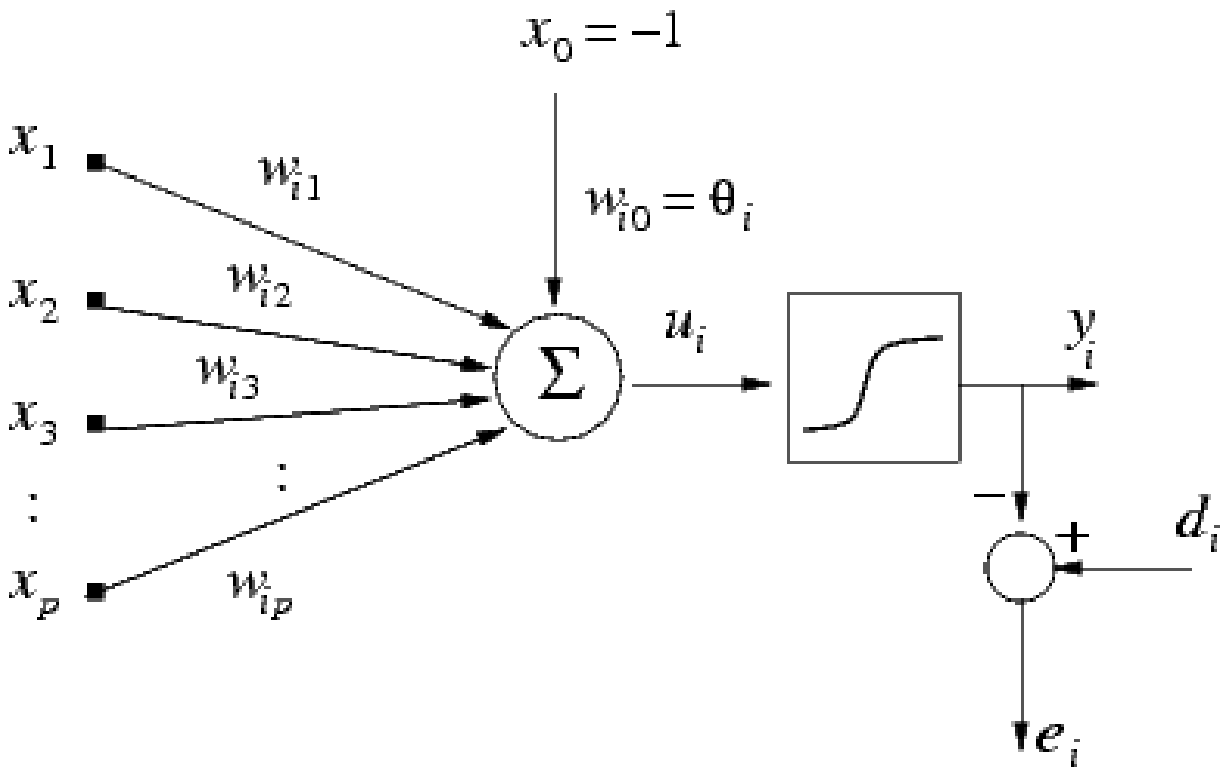
Uma rede MLP com 15 variáveis de entrada (p=15), 20 neurônios na 1a. camada oculta (q_1=20), 10 neurônios na 2a. camada oculta (q_2=10) e 4 neurônios de saída (m=4) é representada como MLP(15,20,10,2).

NOTA 1: A especificação de p e m são ditadas pela forma como o problema é codificado para ser resolvido por uma rede neural.

NOTA 2: As especificações de q_1 e q_2 dependem da complexidade do problema, ou seja, é preciso realizar vários testes até encontrar os valores mais adequados.

8. Rede Perceptron Multicamadas

Um neurônio qualquer da rede MLP, seja oculto ou de saída, é representado genericamente como na figura abaixo.



8. Rede Perceptron Multicamadas

Note que a função de ativação do neurônio M-P, que é do tipo

Degrau (não-linearidade dura ou *hard*)

foi substituída por uma função de ativação do tipo

Sigmoidal (não-linearidade suave ou *soft*).

Assim, a saída deixa de ser uma variável do tipo

ON-OFF (binária [0,1] ou bipolar [-1,+1])

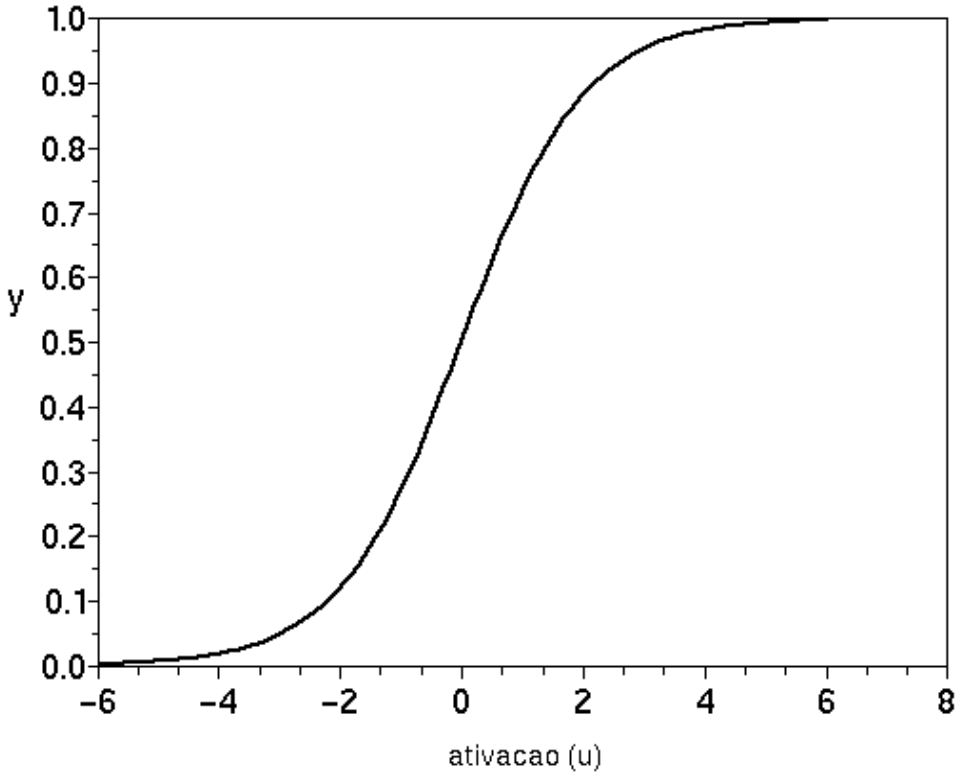
e passa a ser uma variável do tipo

Real ou **Analógica** (qq valor entre [0,1] ou [-1,+1]).

Função de ativação ***Sigmóide Logística***

$$y_i(t) = \frac{1}{1+\exp(-u_i(t))}$$

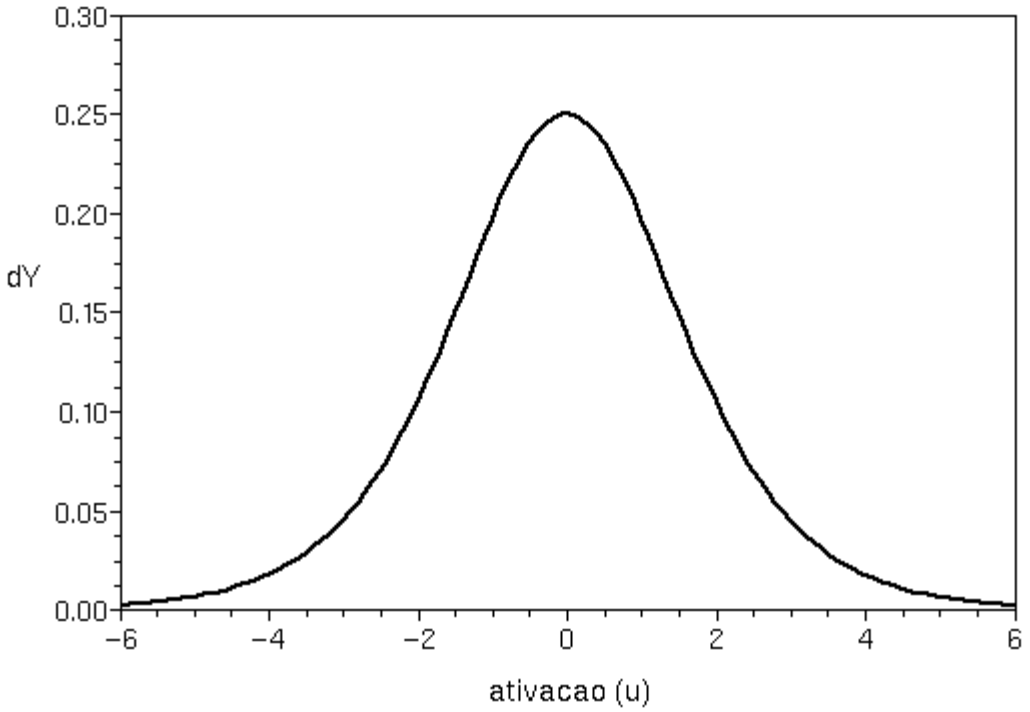
$$y_i(t) \in (0,1)$$



Derivada da ***Sigmóide Logística***

$$y_i'(t) = \frac{d\, y_i(t)}{d\, u_i(t)}$$

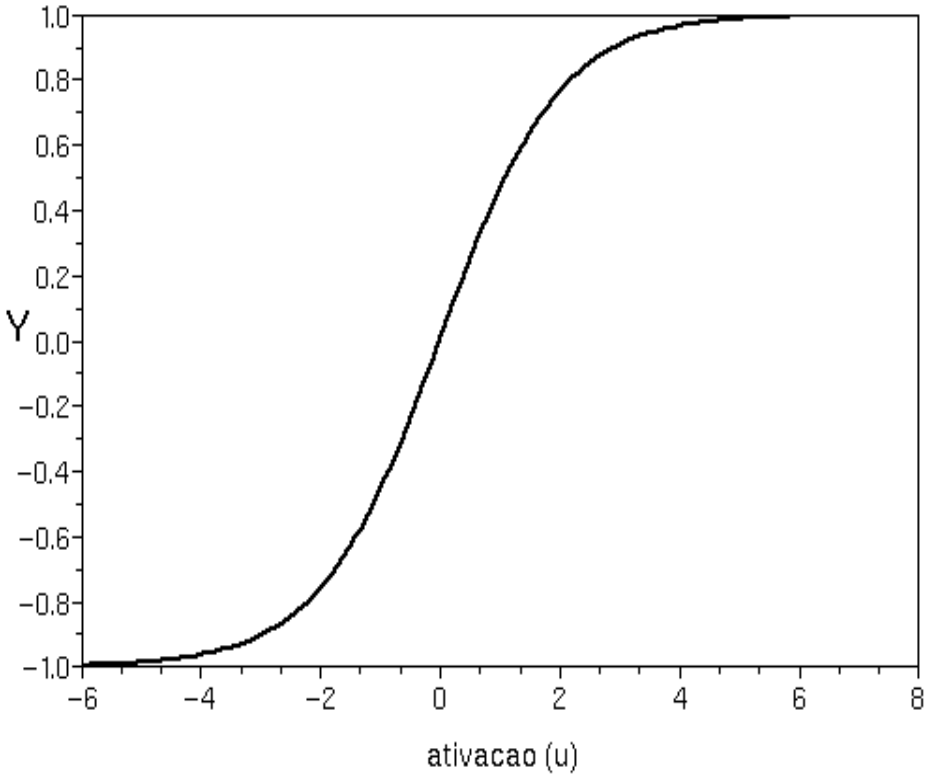
$$y_i'(t) = y_i(t)[1 - y_i(t)]$$



Função de ativação ***Tangente Hiperbólica***

$$y_i(t) = \frac{1 - \exp(-u_i(t))}{1 + \exp(-u_i(t))}$$

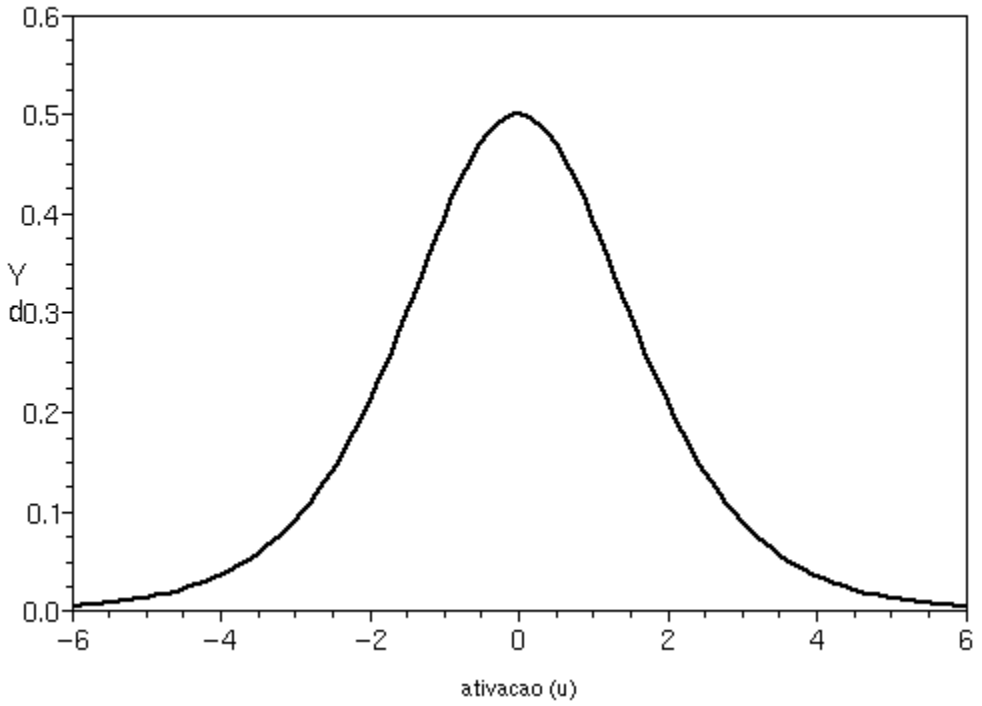
$$y_i(t) \in (-1,1)$$



Derivada da *Tangente Hiperbólica*

$$y_i'(t) = \frac{d\, y_i(t)}{d\, u_i(t)}$$

$$y_i'(t) = 0{,}5\left[1 - y_i^{\,2}(t)\right]$$



Sobre o Uso de Funções de Ativação Sigmoidais

Vantagens: (1) Derivadas fáceis de calcular.

(2) Não-linearidade fraca (trecho central é quase linear)

(3) Interpretação da saída como taxa média de disparo (*mean firing rate*), em vez de simplesmente indicar se o neurônio está ou não ativado (ON-OFF).

Desvantagens:

(1) Elevado custo computacional para implementação em sistemas embarcados devido à presença da função EXP.

$$\exp(x) = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

Funcionamento de uma rede MLP com 1 camada oculta

(1) A ativação do *i*-ésimo neurônio da camada oculta é dada por:

$$u_i = \mathbf{w}_i^T \mathbf{x} = w_{i0} x_0 + w_{i1} x_1 + w_{i2} x_2 + \ldots + w_{ip} x_p, \quad i = 1, \ldots, q_1$$

(2) A saída do *i*-ésimo neurônio da camada oculta é dada por:

$$z_i(t) = \frac{1}{1 + \exp(-u_i(t))}, \quad i = 1, \ldots, q_1$$

ou

$$z_i(t) = \frac{1 - \exp(-u_i(t))}{1 + \exp(-u_i(t))}, \quad i = 1, \ldots, q_1$$

Funcionamento de uma rede MLP com 1 camada oculta

(1) A ativação do *k*-ésimo neurônio de saída é dada por:

$$a_k = \mathbf{m}_k^T \mathbf{z} = m_{k0} z_0 + m_{k1} z_1 + m_{k2} z_2 + \cdots + m_{kq_1} z_{q_1}, \quad k = 1, \ldots, m$$

(2) A saída do *k*-ésimo neurônio de saída é dada por:

$$o_k(t) = \frac{1}{1 + \exp(-a_k(t))}, \quad k = 1, \ldots, m$$

ou

$$o_k(t) = \frac{1 - \exp(-a_k(t))}{1 + \exp(-a_k(t))}, \quad k = 1, \ldots, m$$

Treinamento de uma rede MLP (1 camada oculta)

O k-ésimo neurônio de saída têm acesso à saída desejada, d_k.

Assim, é possível calcular o erro associado a esse neurônio:

$$e_k = d_k - o_k$$

Este erro pode então ser utilizado em uma regra de aprendizagem similar àquela usada pelo algoritmo Perceptron Simples.

$$\mathbf{m}_k(t+1) = \mathbf{m}_k(t) + \eta\, e_k(t)\, o'_k(t)\, \mathbf{z}(t)$$

Onde : $\mathbf{z}(t)$ é o vetor de entrada da camada de saída.

$$o'_k(t) = o_k(t)[1 - o_k(t)] \qquad \text{(p/ sigmóide logística)}$$

$$o'_k(t) = 0{,}5[1 - o_k^2(t)] \qquad \text{(p/ tangente hiperbólica)}$$

Treinamento de uma rede MLP (1 camada oculta)

Contudo, o *i*-ésimo neurônio oculto não tem acesso a uma saída desejada equivalente, d_i.

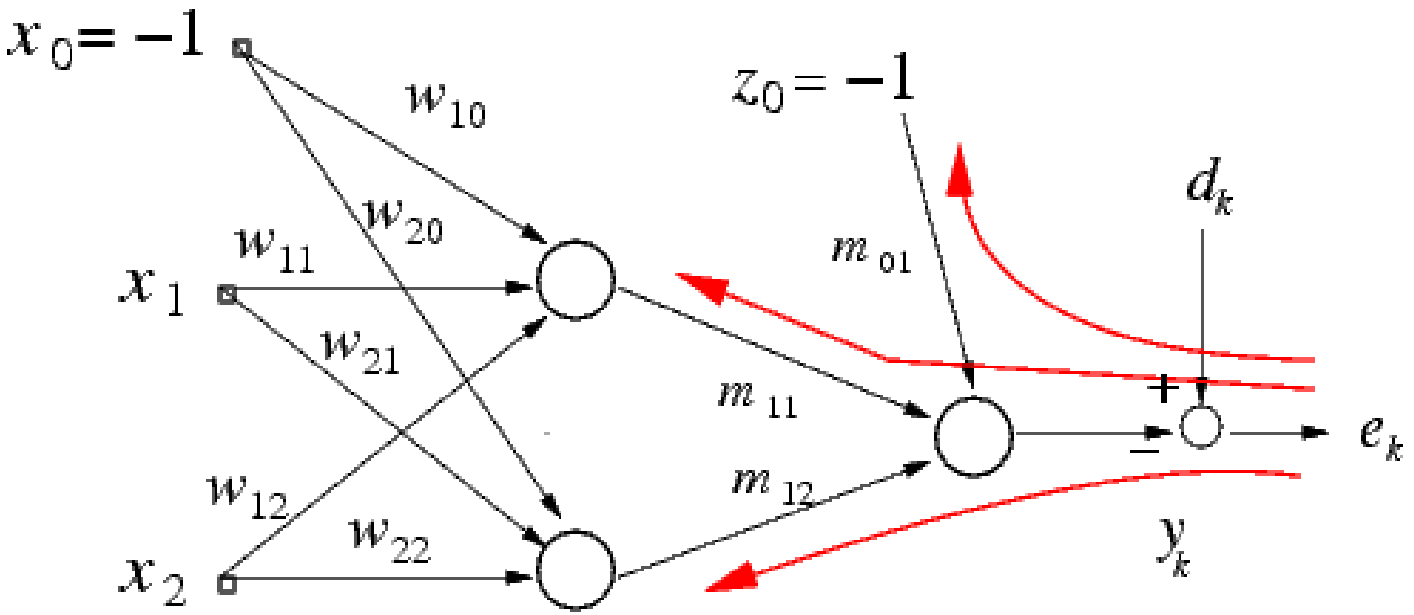
Assim, NÃO é possível calcular o erro associado a esse neurônio.

A saída encontrada pelos pesquisadores foi "inventar" uma espécie de erro para os neurônios ocultos, sem que houvesse a necessidade de uma saída desejada, d_i.

O erro dos neurônios ocultos são obtidos a partir dos erros dos neurônios de saída por meio de uma **projeção no sentido inverso** ao do fluxo de informação convencional.

Treinamento de uma rede MLP (1 camada oculta)

Esta projeção no sentido inverso dos erros de saída é mais conhecida pelo nome de retropropagação dos erros (*Error Backpropagation*).



8. Rede Perceptron Multicamadas

O algoritmo de *backpropagation* é o mais usado para treinar redes MLP, tendo sido proposto por diferentes autores em diferentes épocas.

P. Werbos (1974). "Beyond regression: new tools for prediction and analysis in the behavioral sciences", PhD thesis, Harvard University, Boston, MA.

D. E. Rumelhart, G. E. Hinton, & R. J. Williams (1986). "Learning representations by back-propagating errors". *Nature*, 323:533-536, 1986.

Y. Le Cun, "Learning processes in an asymmetric threshold network", In: *Disordered Systems and Biological Organization* (eds. F. Soulie, E. Bienenstock, and G. Weisbuch, Eds.). Les Houches, France: Springer-Verlag, 1986, pp. 233-340.

Parker, D. (1985). "Learning Logic", *Technical Report* TR-87. Cambridge, MA: Center for Computational Research in Economics and Management Science, MIT.

Regras de Aprendizagem da Rede MLP (1 camada oculta)

Neurônios Ocultos:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \eta\, e_i(t)\, z_i'(t)\, \mathbf{x}(t)$$

Onde: e_i é o erro retroprojetado do i-ésimo neurônio de saída

$$e_i(t) = \sum_{k=1}^{m} m_{ki}\, o_k'(t)\, e_k(t), \qquad i = 1, \ldots, q_1$$

$\mathbf{x}(t)$ é o vetor de entrada da rede.

$z_i'(t) = z_i(t)[1 - z_i(t)]$ (p/ sigmóide logística)

$z_i'(t) = 0{,}5[1 - z_i^2(t)]$ (p/ tangente hiperbólica)

Obtenção Teórica da Regra de Aprendizagem da Rede MLP

Para a rede PS, a regra de aprendizagem foi obtida através de uma análise geométrica do problema.

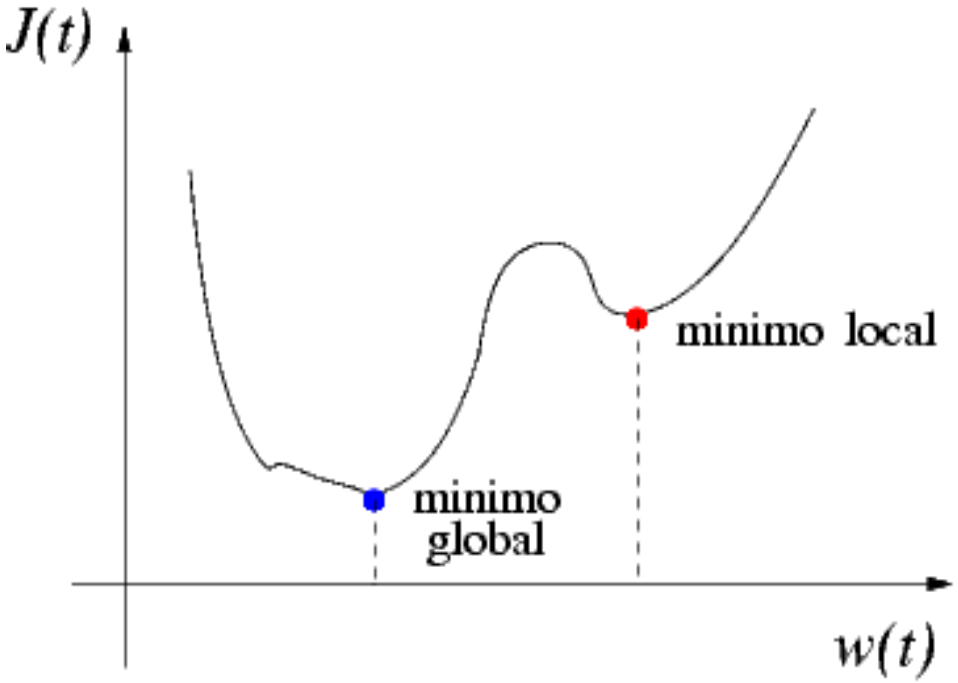
Para a rede MLP vamos obter uma regra de aprendizagem semelhante, a partir da minimização de uma função-custo (ou função objetivo).

Para isso, considere inicialmente que o *erro quadrático instantâneo* para todos os *m* neurônios de saída é dado por:

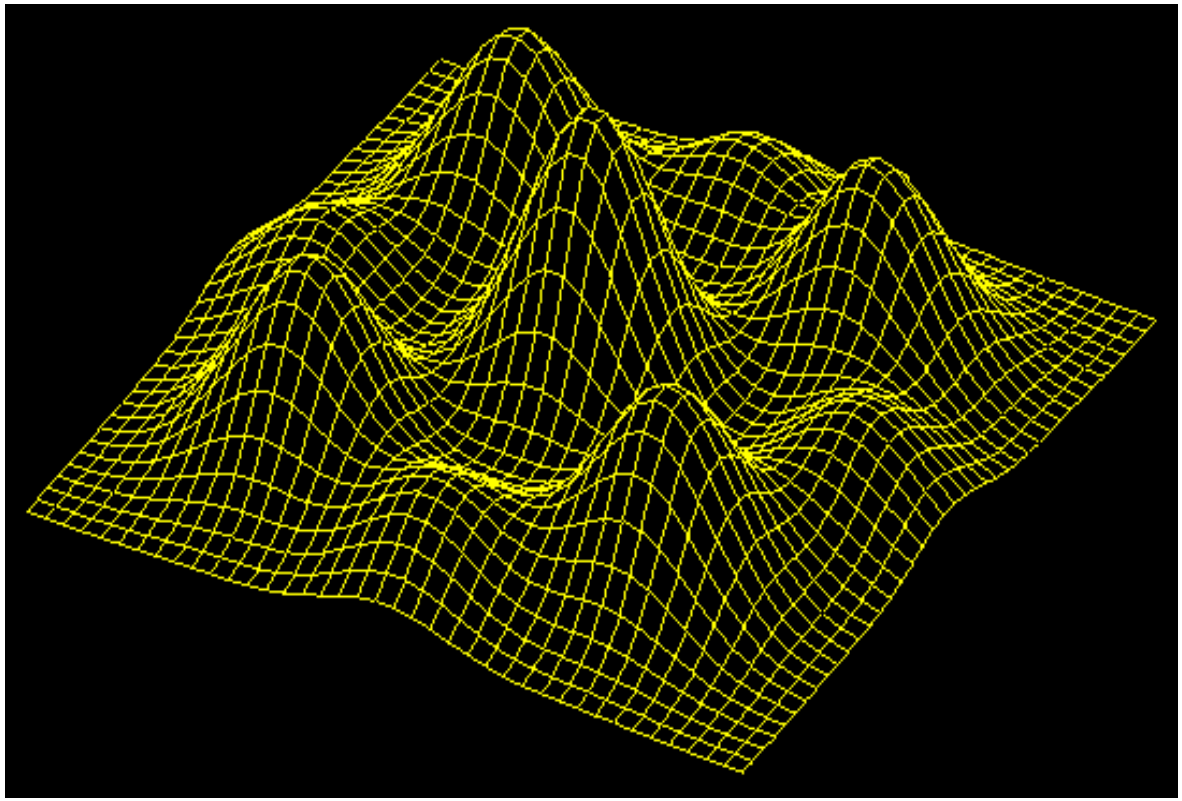
$$J(t)=\frac{1}{2}\sum_{k=1}^{m} e_k^2(t)=\frac{1}{2}\sum_{k=1}^{m}\left(d_k(t)-o_k(t)\right)^2$$

Obtenção da Regra de Aprendizagem da Rede MLP

Gráfico hipotético que ilustra o efeito da não-linearidade na função $J(t)$ para um único neurônio de saída com peso w é mostrado abaixo:



Superfície de busca hipotética em três dimensões



Obtenção da Regra de Aprendizagem da Rede MLP

A função-custo de interesse é o *Erro Quadrático Médio* (EQM), para os N exemplos de treinamento:

$$J(\mathbf{W}) = \frac{1}{N}\sum_{t=1}^{N} J(t) = \frac{1}{2N}\sum_{t=1}^{N}\sum_{k=1}^{m} e_k^2(t)$$

$$= \frac{1}{2N}\sum_{t=1}^{N}\sum_{k=1}^{m}\left(d_k(t) - o_k(t)\right)^2$$

onde $\mathbf{W}$ é o conjunto de todos os parâmetros (pesos e limiares) da rede.

Note que a função $J(\mathbf{W})$ pode ser minimizada ao se minimizar $J(t)$!

Obtenção da Regra de Aprendizagem da Rede MLP

Como a função custo é não-linear, então o processo de minimização deve ser realizado de modo iterativo por meio da seguinte equação recursiva:

$$\mathbf{m}_k(t+1) = \mathbf{m}_k(t) - \eta \frac{\partial J(t)}{\partial \mathbf{m}_k(t)}$$

onde η é passo de aprendizagem ($0 < \eta < 1$).

Note que o segundo termo da equação acima é o incremento imposto ao vetor de pesos $\mathbf{w}_k$ no instante *t*, ou seja

$$\Delta \mathbf{m}_k(t+1) = -\eta \frac{\partial J(t)}{\partial \mathbf{m}_k(t)}$$

Obtenção da Regra de Aprendizagem da Rede MLP

Temos então que calcular a seguinte derivada, também chamada de gradiente da função $J(t)$ na direção do vetor $\mathbf{m}_k(t)$:

$$\frac{\partial J(t)}{\partial \mathbf{m}_k(t)}$$

Para isso, usaremos a regra da cadeia para fatorar esta derivada em vários termos:

$$\frac{\partial J(t)}{\partial \mathbf{m}_k(t)} = \frac{\partial J(t)}{\partial e_k(t)} \frac{\partial e_k(t)}{\partial o_k(t)} \frac{\partial o_k(t)}{\partial a_k(t)} \frac{\partial a_k(t)}{\partial \mathbf{m}_k(t)}$$

8. Rede Perceptron Multicamadas

Vamos calcular cada derivada separadamente:

(1) Se $J(t)=\frac{1}{2}\sum_{k=1}^{m} e_k^2(t)$ Então $\frac{\partial J(t)}{\partial e_k(t)}=e_k(t)$

(2) Se $e_k = d_k - o_k$ Então $\frac{\partial e_k(t)}{\partial o_k(t)}=-1$

(3) Se $o_k(t)=\varphi\left(a_k(t)\right)$ Então $\frac{\partial o_k(t)}{\partial a_k(t)}=\varphi'(t)=o_k'(t)$

(4) Se $a_k(t)=\mathbf{m}_k^T(t)\,\mathbf{z}(t)$ Então $\frac{\partial a_k(t)}{\partial \mathbf{m}_k(t)}=\mathbf{z}(t)$

Juntando novamente cada derivada, chega-se ao seguinte resultado:

$$\frac{\partial J(t)}{\partial \mathbf{m}_k(t)} = \frac{\partial J(t)}{\partial e_k(t)} \frac{\partial e_k(t)}{\partial o_k(t)} \frac{\partial o_k(t)}{\partial a_k(t)} \frac{\partial a_k(t)}{\partial \mathbf{m}_k(t)}$$

$$= e_k(t)(-1)o'(t)\mathbf{z}(t) = -e_k(t)o'(t)\mathbf{z}(t)$$

Assim, obtemos a regra de aprendizagem para os neurônios de saída:

$$\mathbf{m}_k(t+1) = \mathbf{m}_k(t) - \eta \frac{\partial J(t)}{\partial \mathbf{m}_k(t)}$$

$$\mathbf{m}_k(t+1) = \mathbf{m}_k(t) + \eta\, e_k(t) o'_k(t) \mathbf{z}(t)$$

8. Rede Perceptron Multicamadas

Para obter a regra de aprendizagem dos neurônios ocultos adota-se um raciocínio semelhante.

Assim, o processo de minimização deve ser realizado de modo iterativo por meio da seguinte equação recursiva:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) - \eta \frac{\partial J(t)}{\partial \mathbf{w}_i(t)}$$

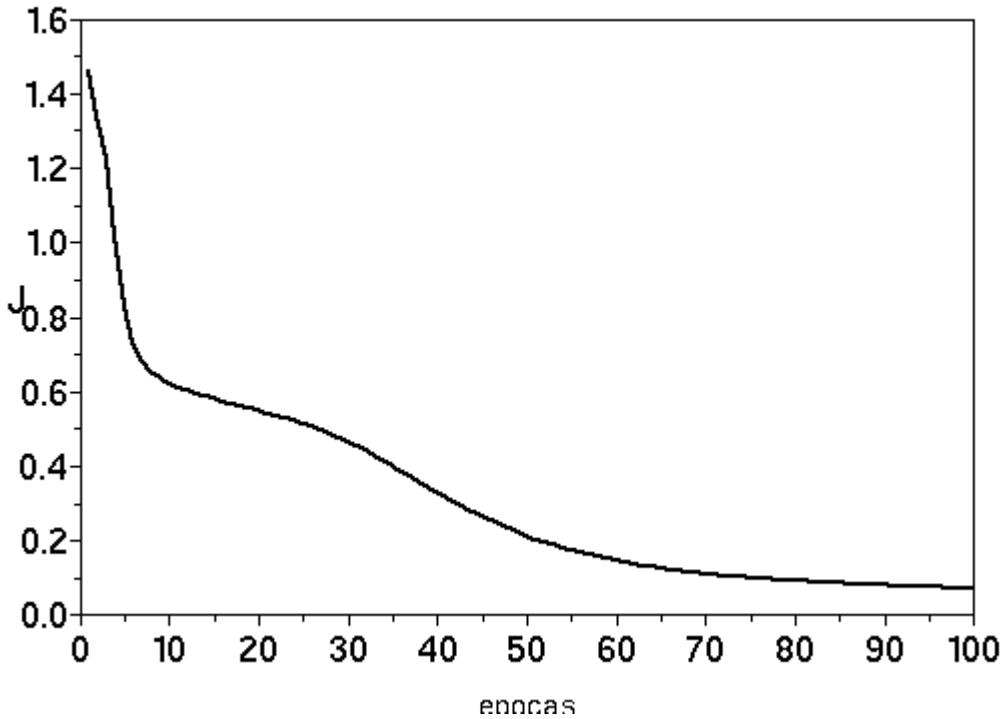
O segredo está em calcular o gradiente da função função custo $J(t)$ agora na direção do vetor $\mathbf{w}_i(t)$, ou seja:

$$\frac{\partial J(t)}{\partial \mathbf{w}_i(t)} = \frac{\partial J(t)}{\partial y_i(t)} \frac{\partial y_i(t)}{\partial u_i(t)} \frac{\partial u_i(t)}{\partial \mathbf{w}_i(t)}$$

(exercício)

Avaliação Gráfica do Treinamento

A avaliação do treinamento é feita através do gráfico de $J(\mathbf{W})$ versus o número de épocas de treinamento, chamado de *curva de aprendizagem*.



A Rede MLP é um Aproximador Universal de Função

Uma rede MLP(p, q_1, m), ou seja, uma rede com uma camada oculta, é capaz de aproximar qualquer função *contínua*, com grau de precisão arbitrário, dado um número *suficientemente* grande de neurônios ocultos com função de ativação sigmoidal.

Uma rede MLP(p, q_1, q_2, m), ou seja, uma rede com duas camadas ocultas, é capaz de aproximar qualquer função *descontínua*, com grau de precisão arbitrário, dado um número *suficientemente* grande de neurônios ocultos com função de ativação sigmoidal em cada camada.

A Rede MLP é um Aproximador Universal de Função

K. Hornik, M. Stinchcombe & H. White (1989). "Multilayer Feedforward Networks are Universal Approximators", *Neural Networks*, vol. 2, no. 5, p. 359--366.

K. Hornik (1991). "Approximation Capabilities of Multilayer Feedforward Networks", *Neural Networks*, vol. 4, no. 2, p. 251--257.

R. Hecht-Nielsen (1987). "Kolmogorov's mapping neural network existence theorem", *Proceedings of the IEEE International Conference on Neural Networks*, pp. 11--14.

J. L. Castro, C. J. Mantas & J. M. Benitez (1987). "Neural networks with a continuous squashing function in the output are universal approximators", *Neural Networks,* vol. 13, no. 6, pp. 561--563.

A Rede MLP é um Aproximador Universal de Função

Isto quer dizer que, por menor que seja o erro quadrático médio exigido para uma boa aproximação do problema, a rede MLP será eventualmente capaz de atingi-lo, desde que o número de neurônios ocultos seja elevado.

Os resultados obtidos nas referências acima são apresentados na forma de *teoremas de existência*, ou seja, eles dizem que existe uma rede MLP que pode aproximar uma certa função, porém não dizem como obtê-la.

Exceção:

V. Kurkova (1992). "Kolmogorov's Theorem and Multilayer Neural Networks", *Neural Networks*, vol. 5, no. 3, p. 501--506.

Papel da Derivada da Função de Ativação na Regra de Aprendizagem

Assumindo uma função de ativação logística, tem-se que

Quando $a_k(t) \to +\infty$, então $o_k(t) \approx 1 \Rightarrow o_k'(t) \approx 0$

e

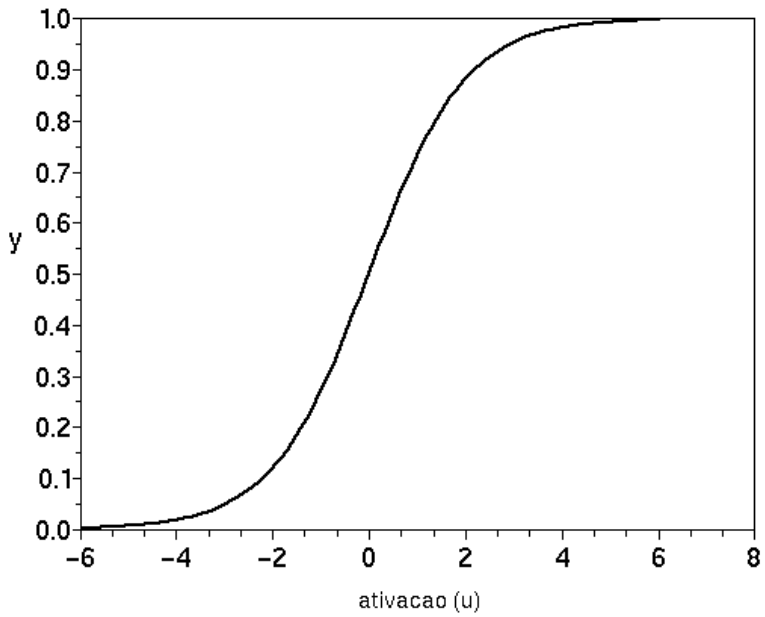
Quando $a_k(t) \to -\infty$, então $o_k(t) \approx 0 \Rightarrow o_k'(t) \approx 0$

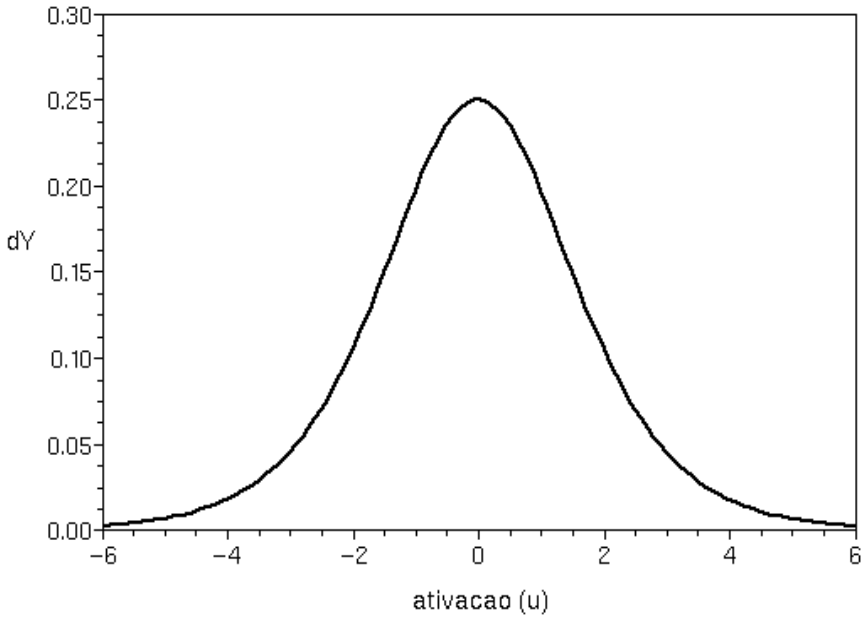
Logo, se $o_k'(t) \approx 0 \Rightarrow \Delta \mathbf{m}_k(t) = \eta\, e_k(t)\, o_k'(t)\, \mathbf{z}(t) \approx 0$

CONCLUSÃO: Quando a ativação é muito alta (ou muito baixa), a saída estará próxima da região de saturação e, assim, o ajuste dos pesos será muito pequeno e o aprendizado mais lento.

8. Rede Perceptron Multicamadas

Este fênomeno recebe o nome de "**paralisia da rede**" e pode ser melhor visualizado colocando-se os gráficos da função de ativação e de sua derivada lado a lado.





8. Rede Perceptron Multicamadas

Para minimizar o efeito da paralisia da rede, é recomendável que a ativação seja mantida em torno da região mais linear da função de ativação, pelo menos no início da fase de aprendizado a fim de acelerá-la.

Dicas para minimizar a paralisia da rede

(1) Iniciar os pesos e limiares com valores pequenos, e.g. na faixa entre [-1/10, +1/10] ou [-1/2,+1/2].

(2) Normalizar as entradas para a faixa [0,+1] ou [-1,+1].

(3) Adicionar 0,05 ao valor da derivada da função de ativação. Assim,

$$\Delta \mathbf{m}_k(t) = \eta\, e_k(t)\,(o'_k(t) + 0{,}05)\,\mathbf{z}(t), \quad \text{onde} \quad e_k(t) = d_k(t) - o_k(t)$$

$$\Delta \mathbf{w}_i(t) = \eta\, e_i(t)\,(z'_i(t) + 0{,}05)\,\mathbf{x}(t), \quad \text{onde} \quad e_i(t) = \sum_{k=1}^{m} m_{ki}\, o'_k(t)\, e_k(t)$$

Determinação dos Neurônios da Camada Oculta

Infelizmente, não existe um método de passo único para determinar o número de neurônios que seja adequado a uma dada tarefa.

Em geral, este número é determinado após alguma experimentação com os dados (*tentativa e erro*).

Por tentativa-e-erro entende-se repetir o processo de treinamento e teste para cada valor especificado para o número de neuônios ocultos.

Algumas técnicas heurísticas podem fornecer um valor inicial para o número de neurônios da camada oculta.

Heurísticas para Especificar o No. de Neurônios Ocultos

(1) Regra do Valor Médio:

$$q_1 = \frac{p+m}{2}$$

(2) Regra da Raiz Quadrada:

$$q_1 = \sqrt{p.m}$$

(3) Regra de Kolmogorov:

$$q_1 = 2p+1$$

(4) Regra de Fletcher-Gloss:

$$2\sqrt{p}+m \leq q_1 \leq 2p+1$$

Heurísticas para Especificar o No. de Neurônios Ocultos

(5) Regra de Baum-Haussler:

$$N > \frac{Z}{\varepsilon}$$

onde

Z é o número de parâmetros da rede MLP(p, q_1, m)

$$Z = (p+1)q_1 + (q_1+1)m$$

N é o tamanho do conjunto de treinamento

ε é o erro tolerado durante o teste.

Exemplo: Se ε = 0,1 (10% de tolerância), então $N > 10\,Z$

Dilema Viés-Variância

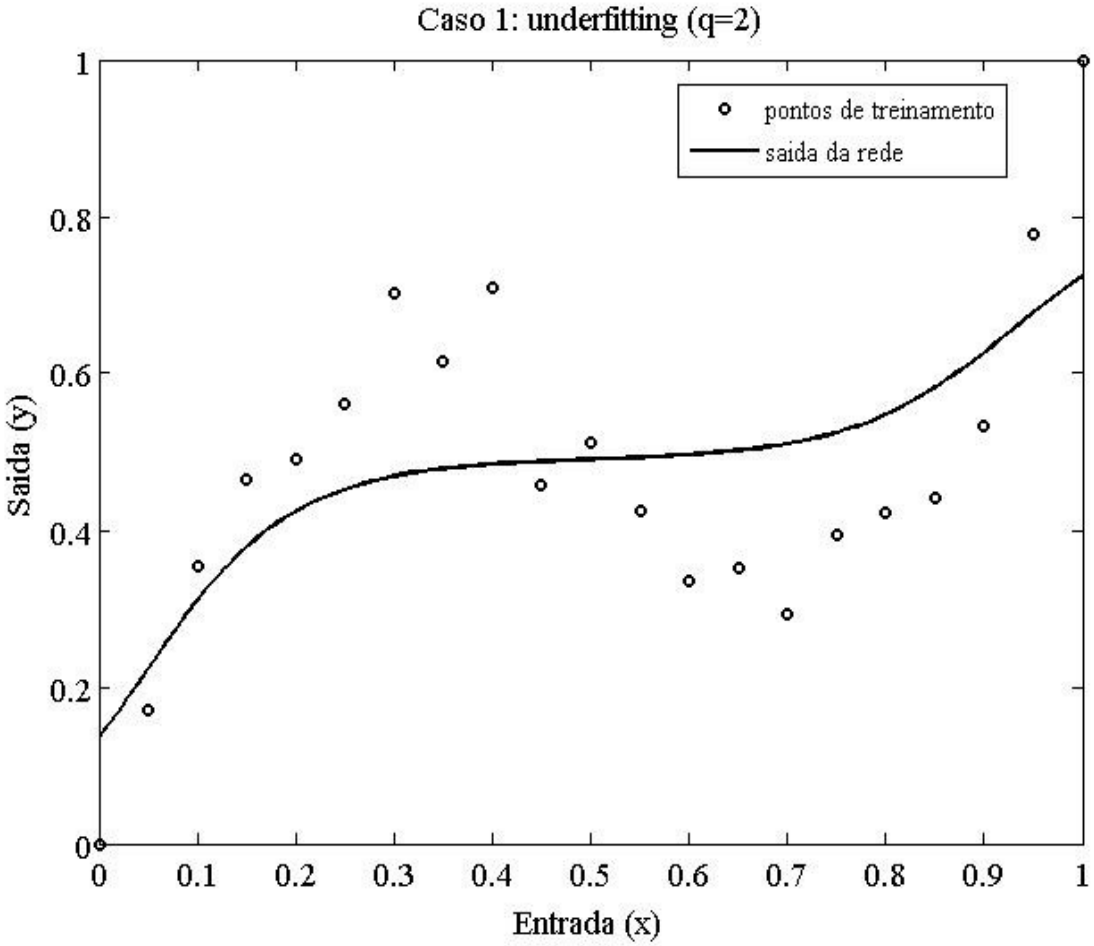
O número de neurônios ocultos deve ser adequadamente determinado, sob pena de o modelo apresentar uma das seguintes situações que levam a um desempenho ruim.

(1) **Underfitting**: a rede neural tem poucos parâmetros ajustáveis e não consegue modelar ou capturar as características específicas dos dados, capturando apenas os detalhes mais gerais. Neste caso, diz-se que a rede generaliza demais.

(1) **Overfitting**: a rede neural tem muitos parâmetros ajustáveis e captura até características muito específicas dos dados, que podem ter sido causadas, e.g. pela presença do ruído. Neste caso, diz-se que a rede generaliza pouco.

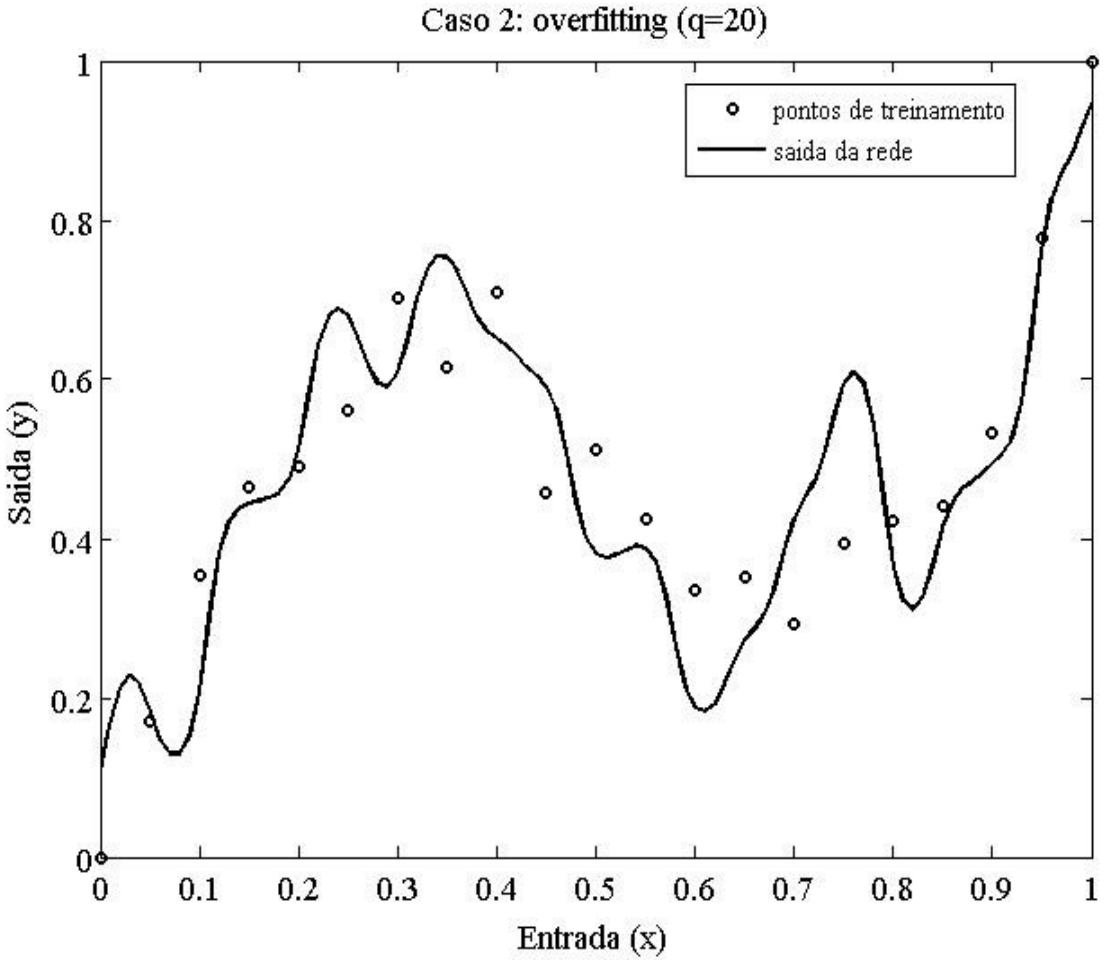
8. Rede Perceptron Multicamadas

Underfitting: Quando a rede generaliza demais, o erro de estimação (viés) é ALTO, porém a variância da saída do modelo é BAIXA.



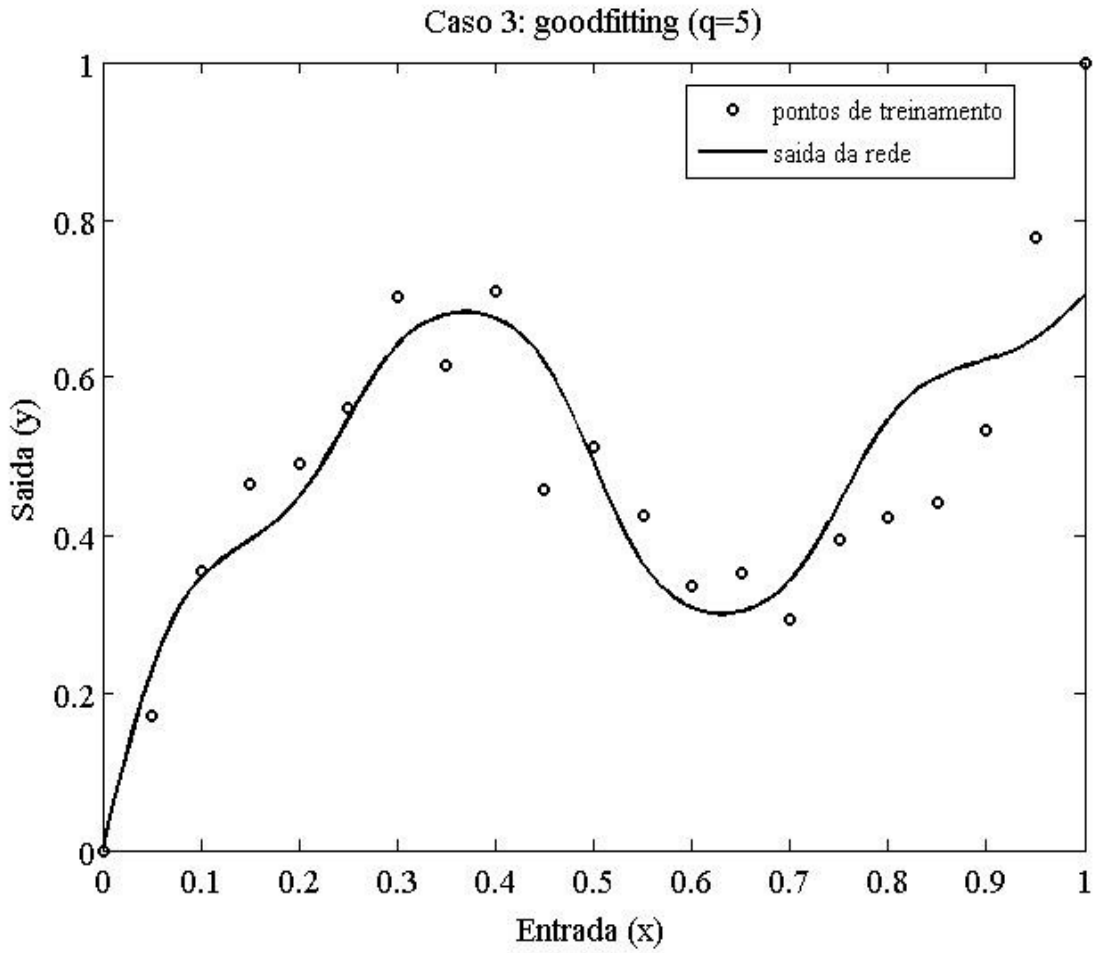
8. Rede Perceptron Multicamadas

Overfitting: Quando a rede generaliza pouco, o erro de estimação (interpolação) é BAIXO, porém a variância do estimador é ALTA.



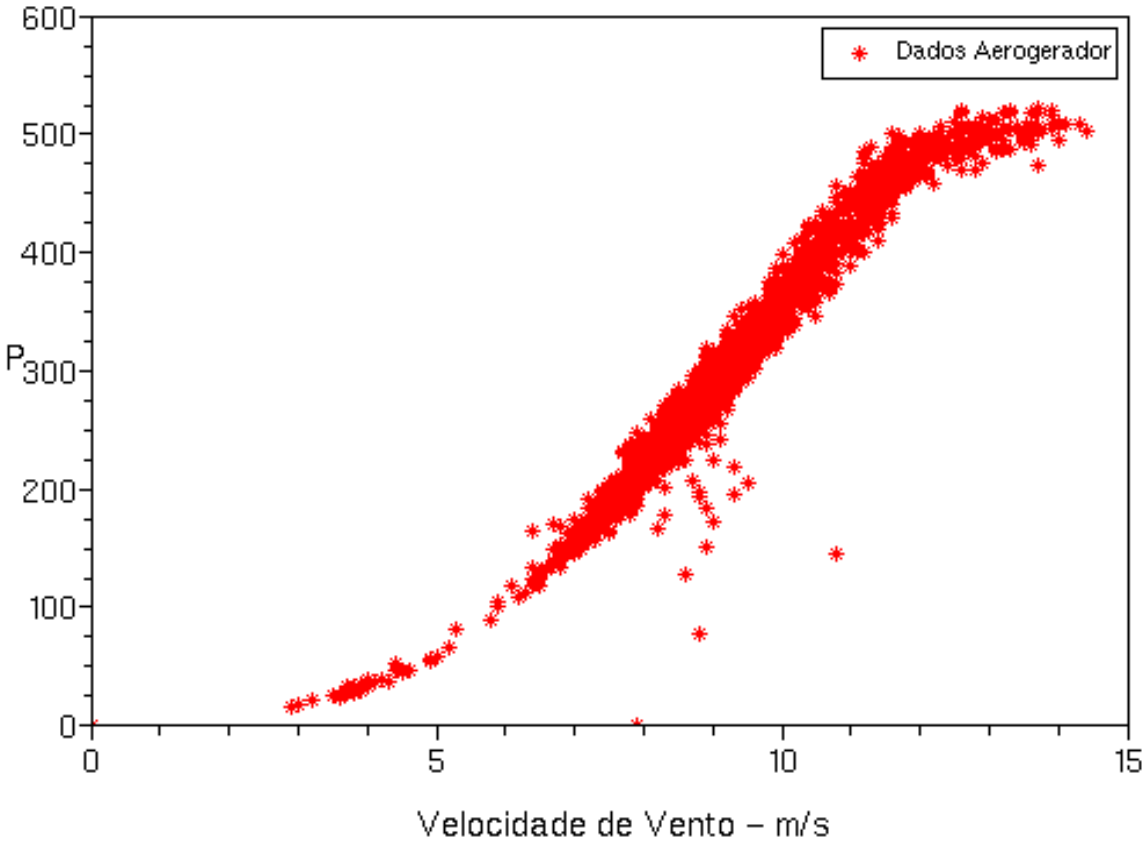
8. Rede Perceptron Multicamadas

Goodfitting: Quando a rede generaliza adequadamente, há um compromisso entre viés (erro de estimação) e a variância do estimador.



8. Rede Perceptron Multicamadas

Aplicação 1: Identificação de um Aerogerador



8. Rede Perceptron Multicamadas

Aplicação 2: Crédito Bancário