

MATB16

Laboratório de Inteligência Artificial

Redes Neurais Artificiais

Tatiane Nogueira Rios
Ricardo Araújo Rios

LabIA
Instituto de Computação - UFBA

Introdução

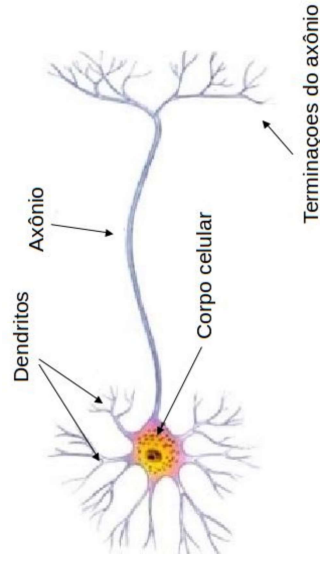
- Vantagens:
 - Aprendizado
 - Generalização
 - Robustez

Introdução

- Redes Neurais Artificiais (RNAs) são modelos matemáticos que se assemelham às estruturas neurais biológicas e que têm capacidade computacional adquirida por meio de aprendizado e generalização (Haykin, 1994)
- São baseadas em modelos abstratos de como pensamos que o cérebro (e os neurônios) funcionam

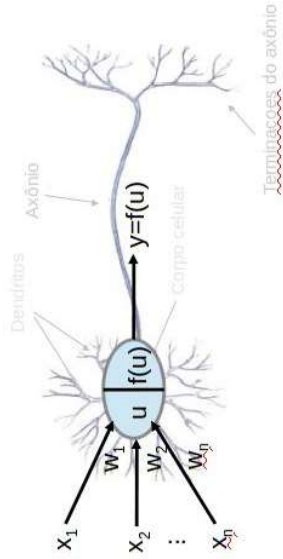
Introdução

- Neurônio Natural



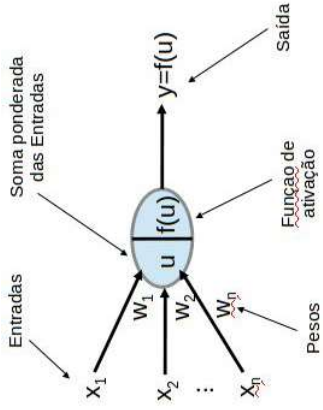
Introdução

- Neurônio artificial (Modelo McCulloch e Pitts - MCP)



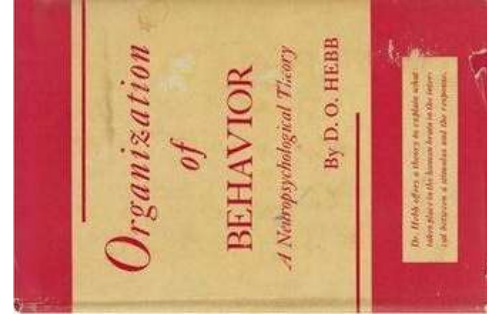
Introdução

- Neurônio artificial



Introdução

Introdução



A LOGICAL CALCULUS OF THE
IMMEDIATE INFERENCES
WALTER S. MCCULLOCH AND WALTER PITTS
Pratt Institute, University of Illinois, Urbana, or MINNESOTA
UNIVERSITY, MINNEAPOLIS, MINN.

Because of the "all-or-none" character of nervous activity, neural excitation can be treated as a series of all-or-none events, just as a human can be regarded as either being in a particular state or not in it. The logic of such nervous activity is easily cast in the form of a calculus which is simple enough to make a case for its use in a model of excitation in the brain. This paper presents just such a model and demonstrates that it is capable of carrying out all the logical operations that are necessary for the control of simple human behavior. It is shown that many particular cases of nervous activity can be interpreted in terms of this model. It is hoped that this model will lead to a better understanding of the logic of nervous activity and that it will be useful in the study of the logic of the brain. Various applications of the calculus are discussed.

1. Introduction

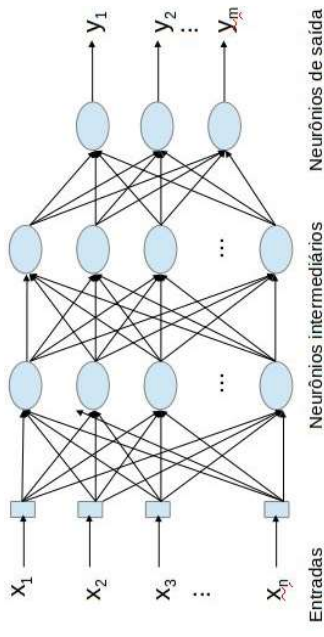
The logical activity of the nervous system is a series of all-or-none events, just as a human can be regarded as either being in a particular state or not in it. The logic of such nervous activity is easily cast in the form of a calculus which is simple enough to make a case for its use in a model of excitation in the brain. This paper presents just such a model and demonstrates that it is capable of carrying out all the logical operations that are necessary for the control of simple human behavior. It is shown that many particular cases of nervous activity can be interpreted in terms of this model. It is hoped that this model will lead to a better understanding of the logic of nervous activity and that it will be useful in the study of the logic of the brain. Various applications of the calculus are discussed.

Introdução

- Redes Neurais Artificiais
 - Várias unidades de processamento (“neurônios”) Interligadas por um grande número de conexões (“sinapses”)

Introdução

- Redes Neurais Artificiais



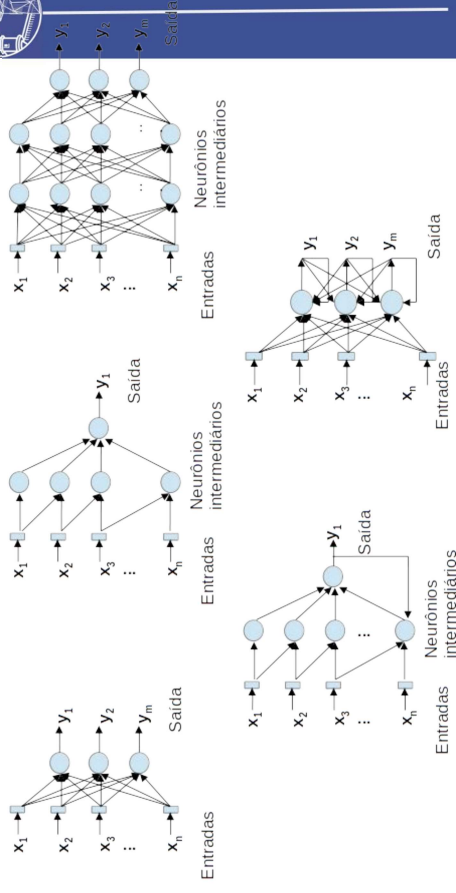
Introdução

- Arquiteturas
 - Restringem o tipo de problema que pode ser tratado pela rede.
 - Redes de camada única: problemas linearmente separáveis
 - Redes recorrentes: problemas que envolvem processamento temporal

Introdução

- Arquiteturas
 - Número de camadas da rede
 - Número de nodos (neurônios) em cada camada
 - Tipo de conexão entre os nodos
 - Topologia da rede

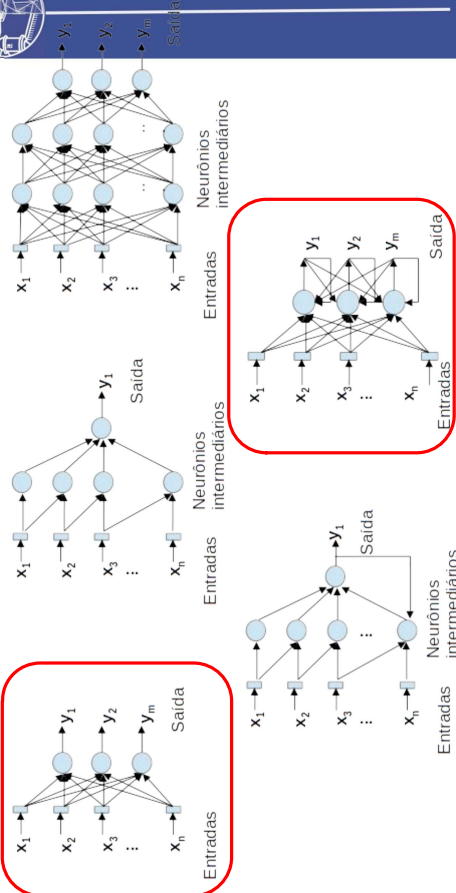
Introdução - Arquiteturas



Introdução - Arquitetura

- Rede de única camada
 - só existe um nó entre qualquer entrada e qualquer saída da rede

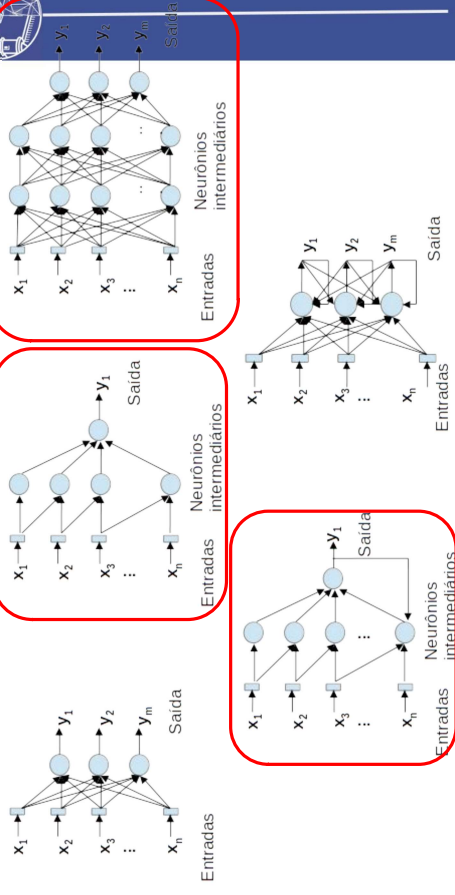
Introdução - Arquiteturas



Introdução - Arquitetura

- Rede de múltiplas camadas
 - Mais de um neurônio entre alguma entrada e alguma saída

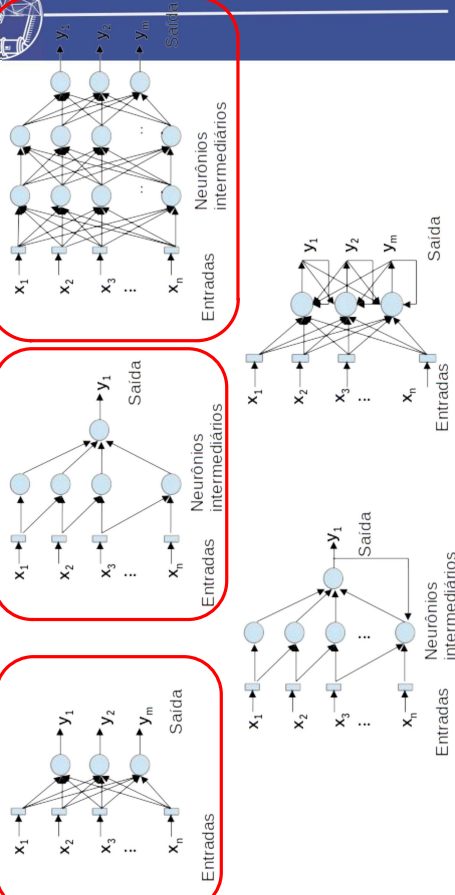
Introdução - Arquiteturas



Introdução - Arquitetura

- Rede feedforward
 - não há "loop", ou seja, os sinais seguem em uma única direção.

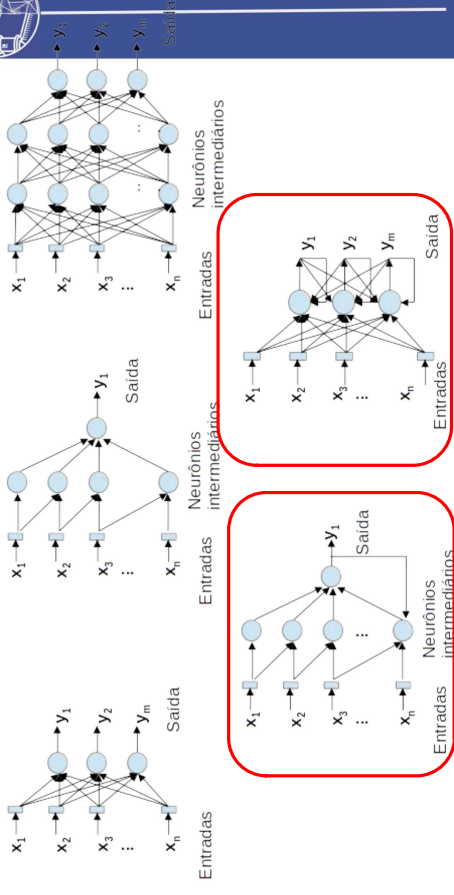
Introdução - Arquiteturas



Introdução - Arquitetura

- Rede feedback
 - a saída de algum neurônio é usada como entrada de outro neurônio

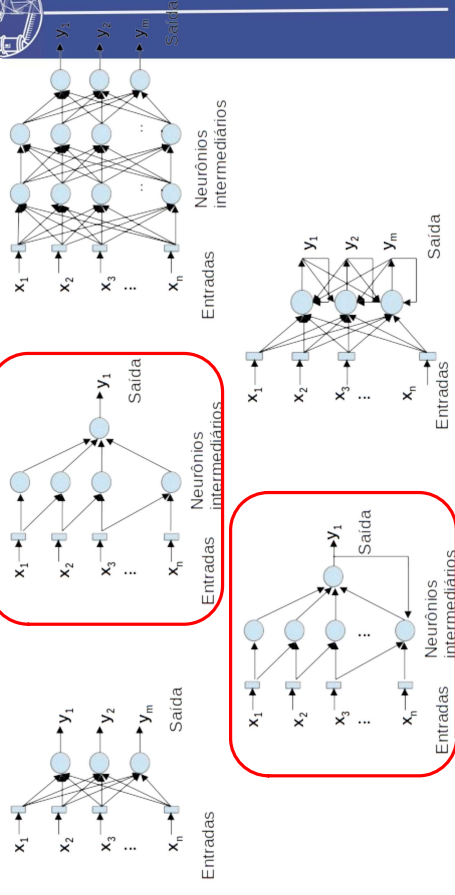
Introdução - Arquiteturas



Introdução - Arquitetura

- Rede fracamente (parcialmente) conectada

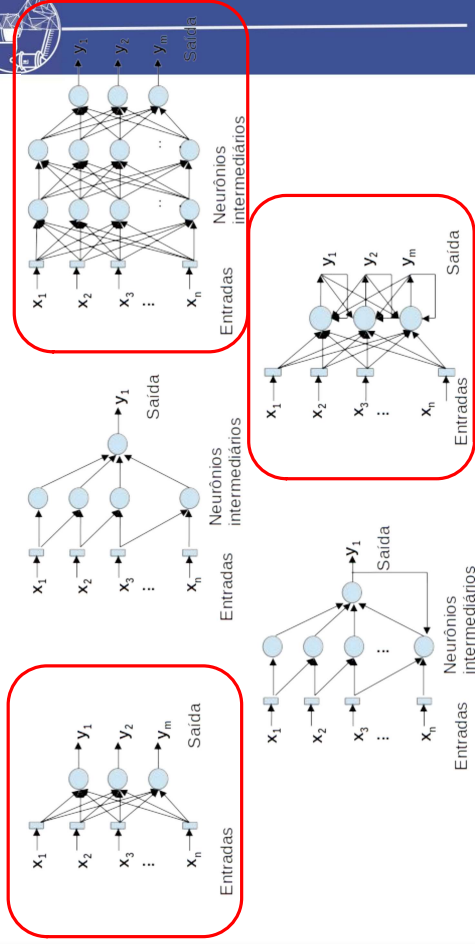
Introdução - Arquiteturas



Introdução - Arquitetura

- Rede fortemente (completamente) conectada

Introdução - Arquiteturas



Aprendizado

- Capacidade de adaptar seus parâmetros como consequência da sua interação com o meio externo.
- Processo iterativo: melhora de desempenho gradativa
- Parâmetros de treinamento:
 - Critério de desempenho
 - Ex: valor do erro quadrático médio das respostas da RNA para um determinado conjunto de dados.
 - Ponto de parada

Aprendizado

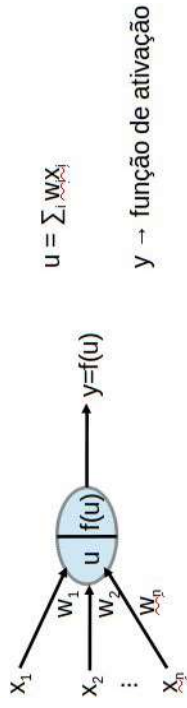
- Aprendizagem é o processo pelo qual os parâmetros de uma rede neural são ajustados por meio de uma forma continuada de estímulo pelo ambiente no qual a rede está operando
- O tipo específico de aprendizagem realizada é definido pela maneira como ocorrem os ajustes realizados nos parâmetros.

Generalização

- Capacidade de dar respostas coerentes para dados não apresentados à RNA previamente durante o treinamento
 - Aprendizado e generalização “andam juntos”

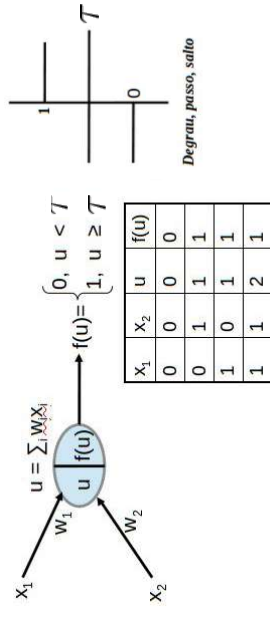
Comportamento Geral

- Aprendizado = ajuste de pesos



Exemplo

- Exemplo: Função lógica OU ($w_1 = w_2 = 1, \tau = 0,5$)



Paradigmas de Aprendizado

- Supervisionado
- Não-supervisionado
- Reforço

Construção da RNA

- Escolher um modelo de neurônio
- Escolher a arquitetura da rede
- Coletar os dados de treinamento
- Projetar uma medida de desempenho ou função objetivo
- Obter a matriz de pesos por meio do aprendizado
 - É a capacidade de aprender que faz uma RNA ser tão poderosa na solução do problema
- Testar a função da rede e desempenho

Perceptron

- Desenvolvido por Rosenblatt, 1958
- Rede mais simples para classificação de padrões linearmente separáveis
- Utiliza modelo de McCulloch-Pitts como neurônio

Perceptron

- Algoritmo
 - Define um bias **b** (θ) com valor inicial tendendo a zero
 - Define um vetor de pesos **w** com valor inicial tendendo a zero
 - Para cada iteração, faça:
 - $u_i = \sum_j w_j x_{ij}$
 - $w_i = w_i + \eta^*(y_i - \hat{y}_i) x_i$
 - $\theta = \theta + \eta^*(y_i - \hat{y}_i)$

Perceptron

- Modelo:
 - Vetor de pesos
 - Bias
 - Erro

Perceptron

- Exercício
 - Dada uma rede do tipo Perceptron formada por um neurônio com três terminais de entrada, utilizando pesos iniciais $w_0 = 0.4$, $w_1 = -0.6$ e $w_2 = 0.6$, $\theta = 0.5$, $\eta = 0.4$ e $\tau = 0$, responda:

Perceptron

- Exercício
 - a) Ensinar a rede a gerar a saída -1 para o padrão 001 e a saída +1 para o padrão 110
 - b) A que classe pertencem os padrões 111, 000, 100 e 011?

Perceptron

- a) Ensinar a rede a gerar a saída -1 para o padrão 001 e a saída +1 para o padrão 110
 - Treinar a rede
 - 001 (-1)
 - $u = 0(0.4) + 0(-0.6) + 1(0.6) + 1(0.5) = 1.1$
 - $f(u) = +1$ ($1.1 > 0$)

Perceptron

- a) Ensinar a rede a gerar a saída -1 para o padrão 001 e a saída +1 para o padrão 110
 - Treinar a rede
 - 001 (-1)
 - $f(u) = +1$ ($1.1 > 0$)
 - $w0 = 0.4 + 0.4(0) (-1 - (+1)) = 0.4$
 - $w1 = -0.6 + 0.4(0) (-1 - (+1)) = -0.6$
 - $w1 = 0.6 + 0.4(1) (-1 - (+1)) = -0.2$
 - $\theta = 0.5 + 0.4 (-1 - (+1)) = -0.3$

Perceptron

- a) Ensinar a rede a gerar a saída -1 para o padrão 001 e a saída +1 para o padrão 110
 - Treinar a rede
 - 110 (+1)
 - $u = 1(0.4) + 1(-0.6) + 0(-0.2) + 1(-0.3) = -0.5$
 - $f(u) = -1$ ($-0.5 < 0$)

Perceptron

- a) Ensinar a rede a gerar a saída -1 para o padrão 001 e a saída +1 para o padrão 110
 - Treinar a rede
 - 110 (+1)
 - $f(u) = -1$ ($-0.5 < 0$)
 - $w0 = 0.4 + 0.4(1) (1 - (-1)) = 1.2$
 - $w1 = -0.6 + 0.4(1) (1 - (-1)) = 0.2$
 - $w1 = -0.2 + 0.4(0) (1 - (-1)) = -0.2$
 - $\theta = -0.3 + 0.4 (1 - (-1)) = 0.5$

Perceptron

- a) Ensinar a rede a gerar a saída -1 para o padrão 001 e a saída +1 para o padrão 110
 - Treinar a rede
 - 001 (-1)
 - $u = 0(1.2) + 0(0.2) + 1(-0.2) + 1(0.5) = 0.3$
 - $f(u) = +1$ ($0.3 > 0$)

Perceptron

- a) Ensinar a rede a gerar a saída -1 para o padrão 001 e a saída +1 para o padrão 110
 - Treinar a rede
 - 001 (-1)
 - $f(u) = +1$
 - $w0 = 1.2$
 - $w1 = 0.2$
 - $w1 = -1.0$
 - $\theta = -0.3$

Perceptron

- a) Ensinar a rede a gerar a saída -1 para o padrão 001 e a saída +1 para o padrão 110
 - Treinar a rede
 - 110 (+1)
 - $u = 1(1.2) + 1(0.2) + 0(-1) + 1(-0.3) = 1.1$
 - $f(u) = +1$

Perceptron

- a) Ensinar a rede a gerar a saída -1 para o padrão 001 e a saída +1 para o padrão 110
 - Treinar a rede
 - 001 (-1)
 - $u = 0(1.2) + 0(0.2) + 1(-1.0) + 1(-0.3) = -1.3$
 - $f(u) = -1$

Perceptron

- a) Ensinar a rede a gerar a saída -1 para o padrão 001 e a saída +1 para o padrão 110
 - Treinar a rede
 - Modelo obtido
 - $w_0 = 1.2$
 - $w_1 = 0.2$
 - $w_1 = -1.0$
 - $\theta = -0.3$

Perceptron

- b) A que classe pertencem os padrões 111, 000, 100 e 011?
 - 111
 - $y = 1(1.2) + 1(0.2) + 1(-1) + 1(-0.3) = 0.1$
 - $f(y) = 1$

Perceptron

- b) A que classe pertencem os padrões 111, 000, 100 e 011?
 - 100
 - $y = 1(1.2) + 0(0.2) + 0(-1) + 1(-0.3) = 0.9$
 - $f(y) = 1$

Perceptron

- b) A que classe pertencem os padrões 111, 000, 100 e 011?
 - 000
 - $y = 0(1.2) + 0(0.2) + 0(-1) + 1(-0.3) = -0.3$
 - $f(y) = -1$

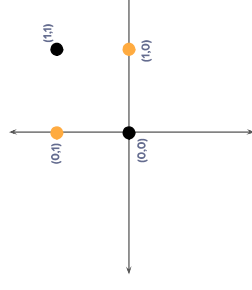
Perceptron

- b) A que classe pertencem os padrões 111, 000, 100 e 011?
 - 011
 - $y = 0(1.2) + 1(0.2) + 1(-1) + 1(-0.3) = -1.1$
 - $f(y) = -1$

MLP

Introdução

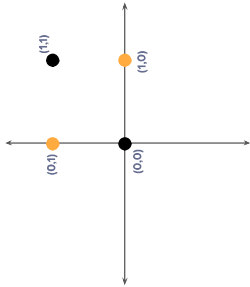
- O problema do XOR





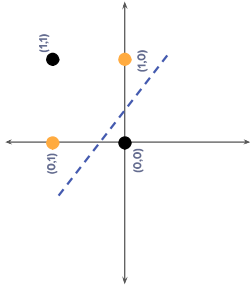
Introdução

- O problema do XOR



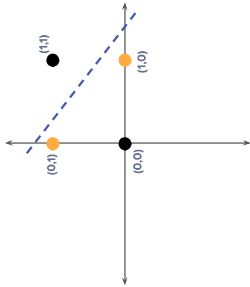
Introdução

- O problema do XOR



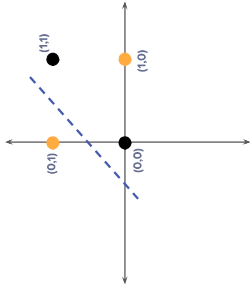
Introdução

- O problema do XOR



Introdução

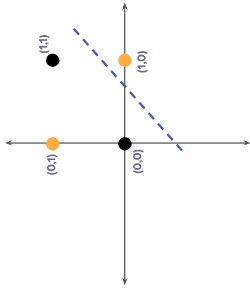
- O problema do XOR





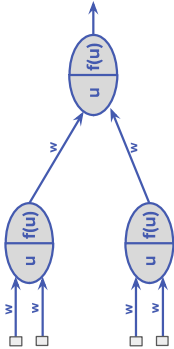
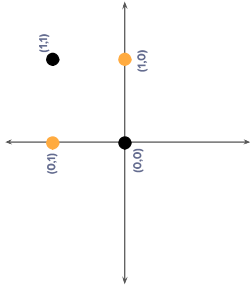
Introdução

- O problema do XOR



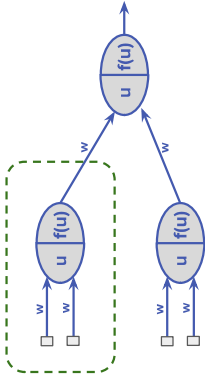
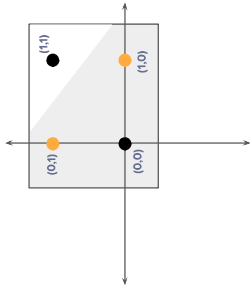
Introdução

- O problema do XOR



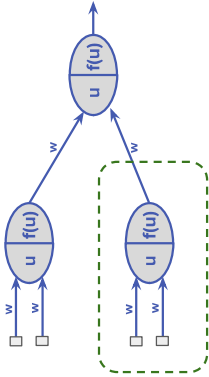
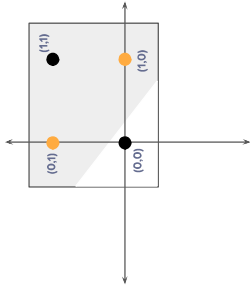
Introdução

- O problema do XOR



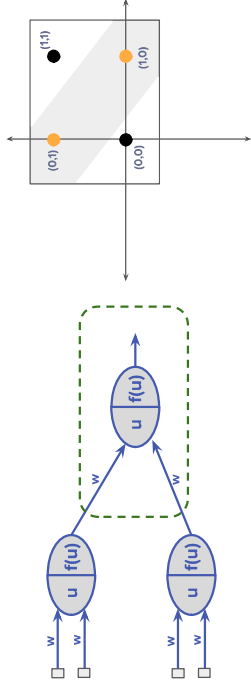
Introdução

- O problema do XOR



Introdução

- O problema do XOR



MLP

- MLPs têm sido aplicados com sucesso em diversos problemas difíceis
- Treinamento:
 - Algoritmo de retropropagação de erro (error back-propagation)
 - Regra de aprendizagem por correção de erro
 - Generalização do algoritmo de mínimo quadrado médio
- Algoritmo básico:
 - Forward: o sinal é aplicado aos nós da rede e seu efeito se propaga camada a camada
 - Back-propagation: pesos são atualizados com base na regra de correção de erro

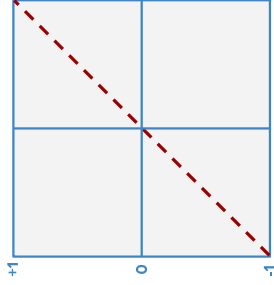
MLP

- MLP possui 3 características principais:
 - Cada neurônio inclui uma função de ativação não-linear. Além disso, a linearidade é suave, ou seja, é diferenciável em qualquer ponto;
 - Contém uma ou mais camadas de neurônios ocultos, capazes de aprender tarefas complexas extraindo progressivamente as características mais significativas dos padrões de entrada;
 - Possui alto grau de conectividade determinado pelas sinapses.

Funções de Ativação

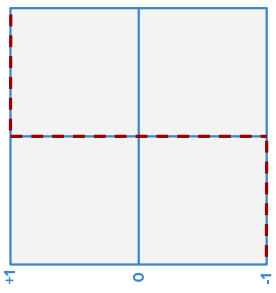
- Identidade

$$f(u) = u$$



Funções de Ativação

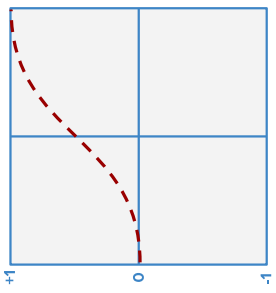
- Degrau



$$f(u) = \text{sign}(u)$$

Funções de Ativação

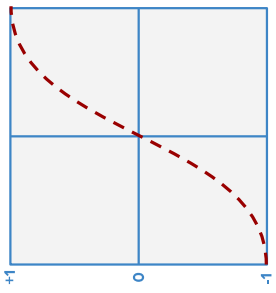
- Sigmoidal



$$f(u) = \frac{1}{1 + e^{-u}}$$

Funções de Ativação

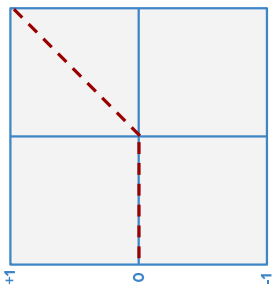
- Tanh:




$$f(u) = \frac{e^{2u} - 1}{e^{2u} + 1}$$

Funções de Ativação

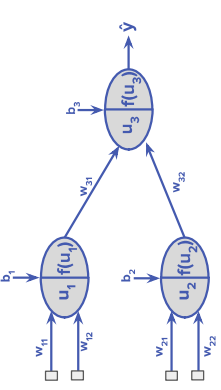
- ReLU




$$f(u) = \max \{u, 0\}$$



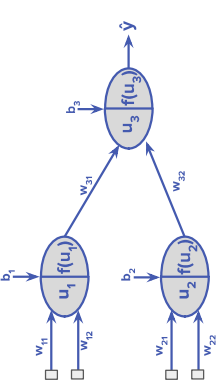
Algoritmo MLP



Forward Step



Algoritmo MLP




Forward Step

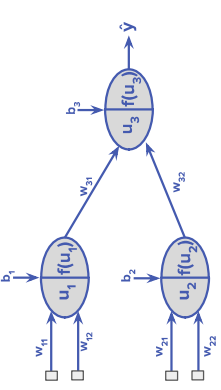
- Pesos sinápticos ficam inalterados em toda rede e a saída de um neurônio j é calculada individualmente:

$$\hat{y}_j = f(u_j)$$

$$u_j = \sum_i x_i w_{ji} + b_j$$




Algoritmo MLP



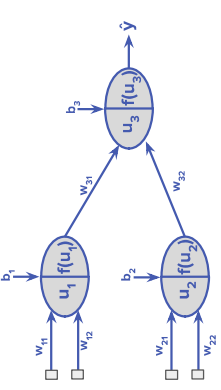
Forward Step

- Para o problema de rotulação [0, 1], utilizaremos uma função de ativação não-linear sigmoidal:

$$f(u_j) = \frac{1}{1 + e^{-au_j}}$$



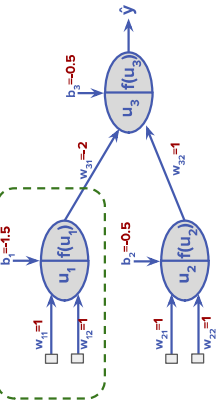
Algoritmo MLP



Forward Step

- Exemplo XOR
 - $(0,0) \rightarrow 0$
 - $(0,1) \rightarrow 1$
 - $(1,0) \rightarrow 1$
 - $(1,1) \rightarrow 0$

Algoritmo MLP



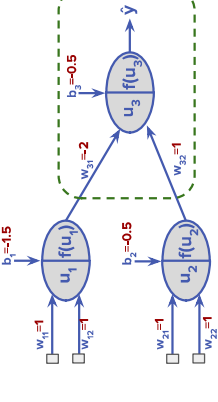
Forward Step

- Exemplo XOR
 - $(0,0) \rightarrow 0$
 - $(0,1) \rightarrow 1$
 - $(1,0) \rightarrow 1$
 - $(1,1) \rightarrow 0$

$u_1 = 0 \cdot 1 + 0 \cdot 1 - 1.5 = -1.5 \mid f(u_1) = 0.1824$

Forward Step

Algoritmo MLP



Forward Step

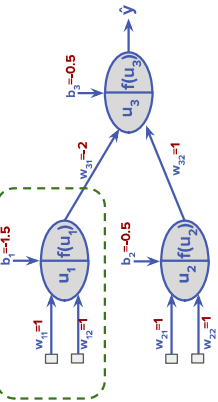
- Exemplo XOR
 - $(0,0) \rightarrow 0$
 - $(0,1) \rightarrow 1$
 - $(1,0) \rightarrow 1$
 - $(1,1) \rightarrow 0$

$u_1 = 0 \cdot 1 + 0 \cdot 1 - 1.5 = -1.5 \mid f(u_1) = 0.1824$
 $u_2 = 0 \cdot 1 + 0 \cdot 1 - 0.5 = -0.5 \mid f(u_2) = 0.3775$

Forward Step

$u_1 = 0 \cdot 1 + 0 \cdot 1 - 1.5 = -1.5 \mid f(u_1) = 0.1824$
 $u_2 = 0 \cdot 1 + 0 \cdot 1 - 0.5 = -0.5 \mid f(u_2) = 0.3775$
 $u_3 = 0 \cdot (-2) + 0 \cdot 1 - 0.5 = -0.5 \mid f(u_3) = 0.38 \rightarrow \hat{y} = 0$

Algoritmo MLP



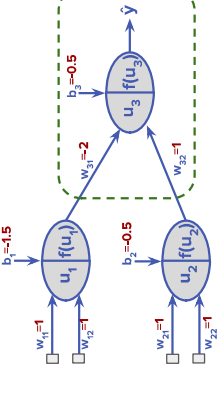
Forward Step

- Exemplo XOR
 - $(0,0) \rightarrow 0$
 - $(0,1) \rightarrow 1$
 - $(1,0) \rightarrow 1$
 - $(1,1) \rightarrow 0$

$u_1 = 0 \cdot 1 + 0 \cdot 1 - 1.5 = -1.5 \mid f(u_1) = 0.1824$

Forward Step

Algoritmo MLP



Forward Step

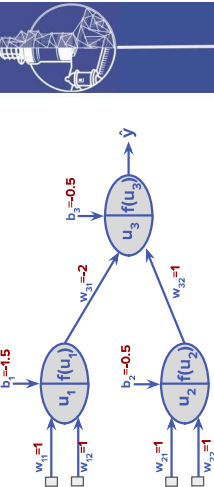
- Exemplo XOR
 - $(0,0) \rightarrow 0$
 - $(0,1) \rightarrow 1$
 - $(1,0) \rightarrow 1$
 - $(1,1) \rightarrow 0$

$u_1 = 0 \cdot 1 + 0 \cdot 1 - 1.5 = -1.5 \mid f(u_1) = 0.1824$
 $u_2 = 0 \cdot 1 + 0 \cdot 1 - 0.5 = -0.5 \mid f(u_2) = 0.3775$

Forward Step

$u_1 = 0 \cdot 1 + 0 \cdot 1 - 1.5 = -1.5 \mid f(u_1) = 0.1824$
 $u_2 = 0 \cdot 1 + 0 \cdot 1 - 0.5 = -0.5 \mid f(u_2) = 0.3775$
 $u_3 = 0 \cdot (-2) + 0 \cdot 1 - 0.5 = -0.5 \mid f(u_3) = 0.38 \rightarrow \hat{y} = 0$

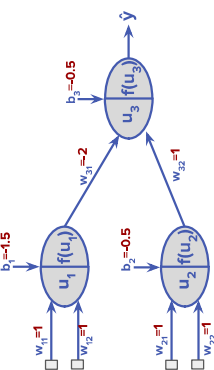
Algoritmo MLP



Forward Step

- Exemplo XOR
 - $(0,0) \rightarrow 0$
 - $(0,1) \rightarrow 1$
 - $(1,0) \rightarrow 1$
 - $(1,1) \rightarrow 0$
- $u_1 = 0 \cdot 1 + 1 \cdot 1 - 1.5 = -0.5 \mid f(u_1) = 0.3775$
 $u_2 = 0 \cdot 1 + 1 \cdot 1 - 0.5 = 0.5 \mid f(u_2) = 0.6225$
 $u_3 = 0 \cdot (-2) + 1 \cdot 1 - 0.5 = 0.5 \mid f(u_3) = 0.6225 \rightarrow \hat{y} = 1$

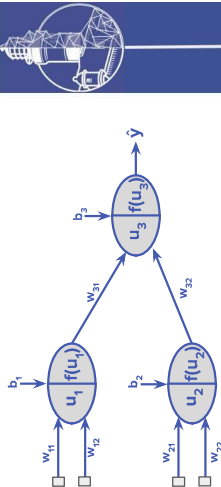
Algoritmo MLP



Forward Step

- Exemplo XOR
 - $(0,0) \rightarrow 0$
 - $(0,1) \rightarrow 1$
 - $(1,0) \rightarrow 1$
 - $(1,1) \rightarrow 0$
- $u_1 = 1 \cdot 1 + 1 \cdot 1 - 1.5 = 0.5 \mid f(u_1) = 0.6225$
 $u_2 = 1 \cdot 1 + 1 \cdot 1 - 0.5 = 1.5 \mid f(u_2) = 0.8176$
 $u_3 = 1 \cdot (-2) + 1 \cdot 1 - 0.5 = -1.5 \mid f(u_3) = 0.1824 \rightarrow \hat{y} = 0$

Algoritmo MLP



Back-propagation Step

- Atualização dos pesos e bias começa da última camada para a primeira, de maneira semelhante ao perceptron

Funções de Loss

- Log Loss / Binary Cross-entropy: classificação binária

$$H_p(y) = -\frac{1}{N} \sum_{i=1}^N y_i \log(P(y_i)) + (1-y_i) \log(1 - P(y_i))$$



Funções de Loss

- Log Loss / Binary Cross-entropy: classificação binária
 - Exemplo de rótulos: (0, 1)
 - Se rótulo esperado é 0

$$H_p(y) = -\frac{1}{N} \sum_{i=1}^N y_i \log(\cancel{P(y_i)}) + (1-y_i) \log(1 - P(y_i))$$



Funções de Loss

- Log Loss / Binary Cross-entropy: classificação binária
 - Exemplo de rótulos: (0, 1)
 - Se rótulo esperado é 1

$$H_p(y) = -\frac{1}{N} \sum_{i=1}^N y_i \log(P(y_i)) + \cancel{(1-y_i) \log(1 - P(y_i))}$$



Funções de Loss

- Exemplo:
$$H_p(y) = -\frac{1}{N} \sum_{i=1}^N y_i \log(P(y_i)) + (1-y_i) \log(1 - P(y_i))$$

Instância	Rótulo esperado	Prob. Predição
1	1	0.94
2	1	0.15
3	0	0.78
	...	
8	0	0.1



Funções de Loss

- Exemplo:
$$H_p(y) = -\frac{1}{N} \sum_{i=1}^N y_i \log(P(y_i)) + (1-y_i) \log(1 - P(y_i))$$

Instância	Rótulo esperado	Prob. Predição
1	1	0.94
2	1	0.15
3	0	0.78
	...	
8	0	0.1

$$H_p(y_1) = -1 \log(0.94) + (1-1) \log(1-0.94) = 0.1246$$

Funções de Loss

- Exemplo:

$$H_p(y) = -\frac{1}{N} \sum_{i=1}^N y_i \log (P(y_i)) + (1-y_i) \log (1 - P(y_i))$$

Instância	Rótulo esperado	Prob. Predição
1	1	0.94
2	1	0.15
3	0	0.78
	...	
8	0	0.1

$$H_p(y_1) = -1 \log (0.94) + (1-1) \log (1 - 0.94) = 0.1246$$

$$H_p(y_2) = -1 \log (0.15) + (1-1) \log (1 - 0.15) = 0.8230$$

$$H_p(y_3) = -0 \log (0.78) + (1-0) \log (1 - 0.78) = 0.6676$$

$$H_p(y_8) = -0 \log (0.1) + (1-0) \log (1 - 0.1) = 0.4158$$

Funções de Loss

- Exemplo:

$$H_p(y) = -\frac{1}{N} \sum_{i=1}^N y_i \log (P(y_i)) + (1-y_i) \log (1 - P(y_i))$$

Instância	Rótulo esperado	Prob. Predição
1	1	0.94
2	1	0.15
3	0	0.78
	...	
8	0	0.1

$$H_p(y_1) = -1 \log (0.94) + (1-1) \log (1 - 0.94) = 0.1246$$

$$H_p(y_2) = -1 \log (0.15) + (1-1) \log (1 - 0.15) = 0.8230$$

$$H_p(y_3) = -0 \log (0.78) + (1-0) \log (1 - 0.78) = 0.6676$$

$$H_p(y_8) = -0 \log (0.1) + (1-0) \log (1 - 0.1) = 0.4158$$

Funções de Loss

- Exemplo:

$$H_p(y) = -\frac{1}{N} \sum_{i=1}^N y_i \log (P(y_i)) + (1-y_i) \log (1 - P(y_i))$$

Instância	Rótulo esperado	Prob. Predição
1	1	0.94
2	1	0.15
3	0	0.78
	...	
8	0	0.1

$$H_p(y_1) = -1 \log (0.94) + (1-1) \log (1 - 0.94) = 0.1246$$

$$H_p(y_2) = -1 \log (0.15) + (1-1) \log (1 - 0.15) = 0.8230$$

$$H_p(y_3) = -0 \log (0.78) + (1-0) \log (1 - 0.78) = 0.6676$$

$$H_p(y_8) = -0 \log (0.1) + (1-0) \log (1 - 0.1) = 0.4158$$

Funções de Loss

- Log Loss / Categorical Cross-entropy
 - Utilizado quando o problema é caracterizado por mais de uma classe
 - N é o número de instâncias
 - M é o número de classes

$$LL_p(y) = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^M y_{ij} \log P(y_{ij})$$

Referências

- RUSSEL, S.; NORVIK, P. Inteligência Artificial. Editora Campus, 2013, Capítulos 12, 13 e 14.
- Faceli et al., Inteligência Artificial – Uma Abordagem de Aprendizado de Máquina, LTC, 2015.
- HAYKIN, Simon; NETWORK, Neural. A comprehensive foundation. Neural networks, v. 2, n. 2004, p. 41, 2004.

