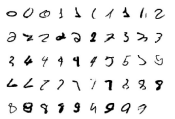
Aprendizado de Máquina

Redes Neurais Artificiais

André C. P. L. F. de Carvalho
ICMC-USP

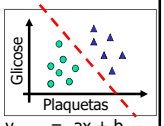
Principais tópicos

- Técnicas geométricas
- Discriminante linear
- Redes neurais
- Arquitetura e aprendizado de redes neurais
- Rede perceptron
- Rede MLP



Discriminante linear

- Busca modelo que melhor se ajuste aos dados
  - Representação matemática
    - Dois atributos preditivos
      - Fronteira de decisão = reta (hiperplano para > 2)
    - Classificação ou regressão
    - Função discriminante
      - Combinação linear dos atributos preditivos
        - Soma ponderada
      - Como definir valores dos pesos?



y = ax + b
f(x) = ax + b
f(x) = -2x +15

Função de classificação:

$$classe(x) = \begin{cases} +1 \text{ se } f(x) + 2p - 15 \geq 0 \\ -1 \text{ se } f(x) + 2p - 15 < 0 \end{cases}$$

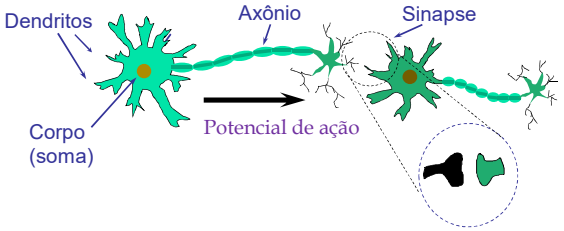
$f(x) = w_0 + w_1x_1$
$f(x) = w_0 + w_1x_1 + w_2x_2 + \ldots$

Redes Neurais

- Sistemas distribuídos inspirados no cérebro humano
  - Redes são compostas por várias unidades de processamento ("neurônios")
    - Interligadas por um grande número de conexões ("sinapses")
- Bom desempenho preditivo em várias aplicações

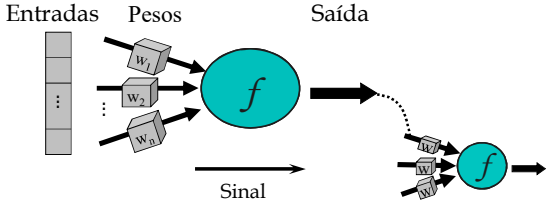
Neurônio natural

- Um neurônio simplificado:



Neurônio artificial

- Modelo de um neurônio abstrato



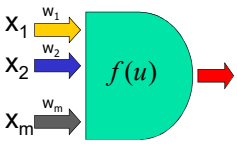
Conceitos básicos

- Principais aspectos das RNA
  - Arquitetura
    - Unidades de processamento (neurônios)
    - Conexões (sinapses)
    - Topologia
  - Aprendizado
    - Algoritmos
    - Paradigmas

Unidades de processamento

- Funcionamento
  - Recebem entradas de conjunto de unidades A
  - Aplicam função de ativação sobre entradas
  - Enviam resultado para saída ou conjunto de unidades B
- Entrada total

$$u = \sum_{i=1}^{m} x_i w_i$$

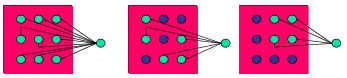


Conexões

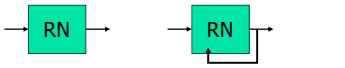
- Definem como neurônios estão interligados
- Codificam conhecimento da rede
- Tipos de conexões:
  - Excitatória: $(w_{ik}(t) > 0)$
  - Inibitória: $(w_{ik}(t) < 0)$

Topologia

- Número de camadas
- Cobertura das conexões



- Arranjo das conexões



Algoritmo de aprendizado

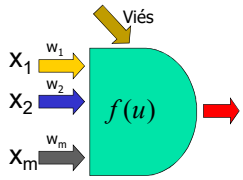
- Conjunto de regras que define como ajustar os parâmetros da rede
- Principais formas de ajuste
  - Correção de erro
  - Hebbiano
  - Competitivo
  - Termodinâmico (Boltzmann)

Paradigma de aprendizado

- Definido pelas informações externas que a rede recebe durante seu aprendizado
  - Principais abordagens
    - Supervisionado
    - Não supervisionado
      - Semi-supervisionado
      - Aprendizado ativo
    - Reforço
    - Híbrido

Perceptron

- Primeira rede - Rosemblat, 1958
  - Modelo de neurônio de McCulloch-Pitts
- Treinamento
  - Supervisionado
  - Correção de erro
    - $w_i(t) = w_i(t-1) + \Delta w_i$
      - $\Delta w_i = \eta x_i \delta$
      - $\Delta w_i = \eta x_i (y - f(\mu))$
- Teorema de convergência



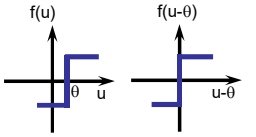
Perceptron

- Resposta / saída da rede
  - Aplica função de ativação limiar sobre soma total de entrada recebida por um neurônio

$$u = \sum_{i=1}^{m} x_i w_i$$

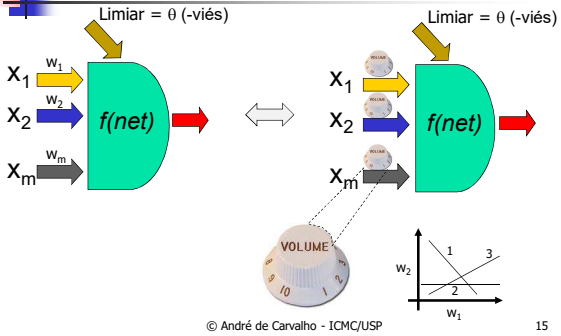
$$f(u) = \begin{cases} +1 \text{ if } u \geq \theta \\ -1 \text{ if } u < \theta \end{cases}$$

$$net = \sum_{i=0}^{m} x_i w_i$$



f(u-θ) = sinal (u-θ)
f(net) = f(u-θ)

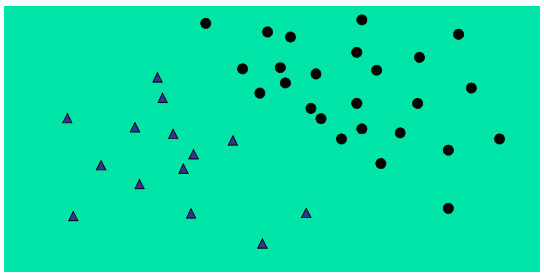
Treinamento



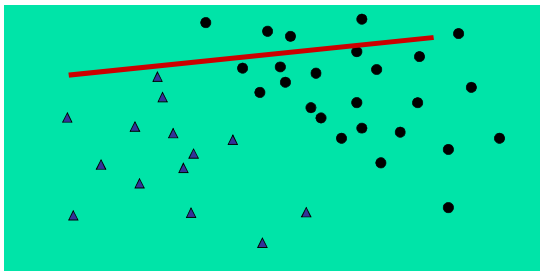
Algoritmo de treinamento

1 Iniciar peso de cada conexão com o valor 0
2 Repita
Para cada par de treinamento (X, y)
Calcular a saída f(X)
Se (y ≠ f(X))
Então
Atualizar pesos do neurônio
Até condição de parada

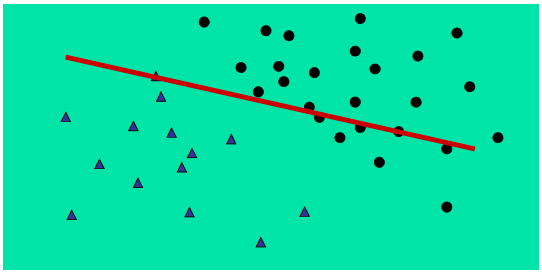
Treinamento modificando fronteiras



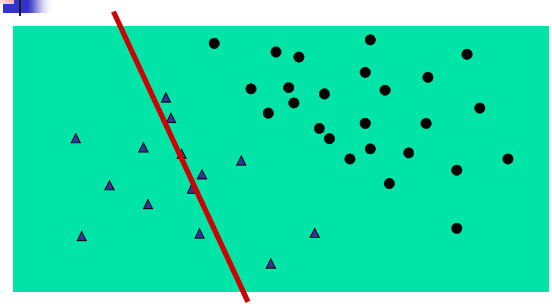
Treinamento modificando fronteiras



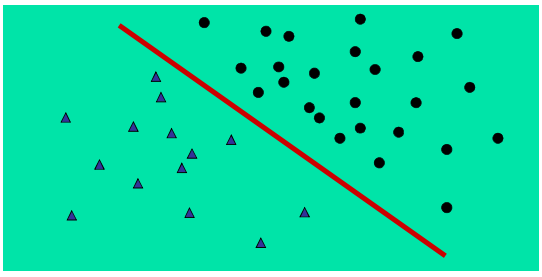
Treinamento modificando fronteiras



Treinamento modificando fronteiras



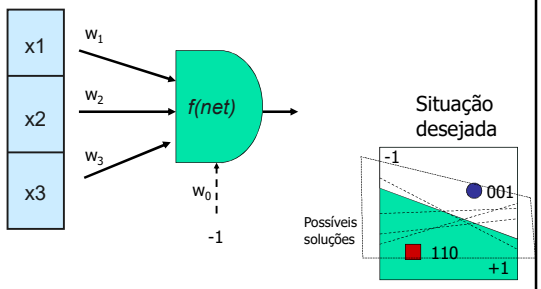
Treinamento modificando fronteiras



Exemplo

- Dada uma rede Perceptron com:
  - Três entradas, pesos $w_1 = 0.4$, $w_2 = -0.6$ e $w_3 = 0.6$, e limiar $\theta = 0.5$:
    - Ensinar a rede com os exemplos (001, -1) e (110, +1)
      - Utilizar taxa de aprendizado $\eta = 0.4$
    - Predizer a classe dos exemplos: 111, 000, 100 e 011

Exemplo



Exemplo (treinamento)

Treinar a rede

$x_0, w_0 (\theta), w_1, w_2, w_3 = -1, 0.5, 0.4, -0.6, 0.6)$

a.1) Para o exemplo 001 (y = -1)

Passo 1: definir a saída da rede (Σxw)

$u\text{-}\theta = -1(0.5) + 0(0.4) + 0(-0.6) + 1(0.6) = 0.1$

$f(net) = +1$ (uma vez $0.1 \geq 0$)

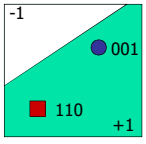
Passo 2: atualizar pesos ($y \neq f(net)$)

$w_0 = 0.5 + 0.4(-1)(-1 - (+1)) = 1.3$

$w_1 = 0.4 + 0.4(0)(-1 - (+1)) = 0.4$

$w_2 = -0.6 + 0.4(0)(-1 - (+1)) = -0.6$

$w_3 = 0.6 + 0.4(1)(-1 - (+1)) = -0.2$



$w_i(t) = w_i(t-1) + \eta x_i(y - f(x))$

Exemplo (teste)

- Utilizar a rede treinada para classificar os exemplos 111, 000, 100 e 011
  - Pesos aprendidos: 0.4, 1.2, 0.2 e -0.2

b.1) Para o exemplo 111

$u\text{-}\theta = -1(0.4) + 1(1.2) + 1(0.2) + 1(-0.2) = 0.6$

f(net) = +1 (porque $0.6 \geq 0$)) $\Rightarrow$ classe +1

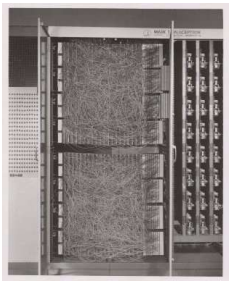
b.2) Para o exemplo 000

$u\text{-}\theta = -1(0.4) + 0(1.2) + 0(0.2) + 0(-0.2) = -0.4$

f(net) = -1 (porque $-0.4 < 0$) $\Rightarrow$ classe -1

Perceptron em hardware

- Primeira implementação:
  - Mark I Perceptron
  - Cornell Aeronautical Laboratory



Exercício

- Seja o seguinte cadastro de pacientes:

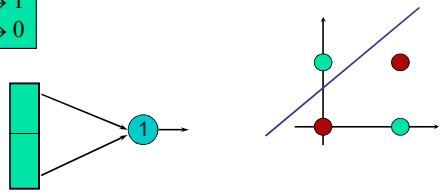
Nome	Febre	Enjôo	Manchas	Dores	Diagnóstico
João	sim	sim	pequenas	sim	doente
Pedro	não	não	grandes	não	saudável
Maria	não	sim	pequenas	não	saudável
José	sim	sim	grandes	sim	doente
Ana	sim	não	pequenas	sim	saudável
Leila	não	não	grandes	sim	doente

Exercício

- Ensinar uma rede do tipo Perceptron a distinguir:
  - Pacientes potencialmente saudáveis
  - Pacientes potencialmente doentes
- Testar a rede para novos casos
  - (Luis, não, não, pequenas, sim)
  - (Laura, sim, sim, grandes, sim)

Problemas com Perceptron

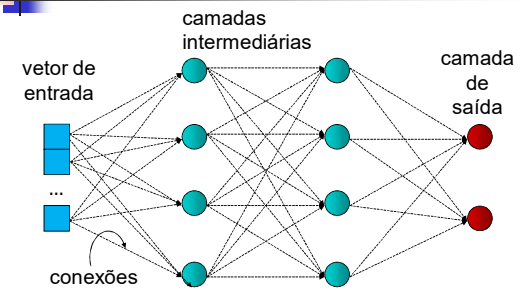
$0, 0 \rightarrow 0$
$0, 1 \rightarrow 1$
$1, 0 \rightarrow 1$
$1, 1 \rightarrow 0$



Rede Multi-Layer Perceptron

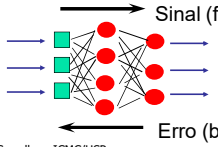
- Arquitetura de RNA mais utilizada
  - Uma ou mais camadas intermediárias de neurônios
- Funcionalidade (teórica)
  - Uma camada intermediária: qualquer função contínua ou Booleana
  - Duas camadas intermediárias: qualquer função
- Originalmente treinada com o algoritmo *backpropagation*

MLP e backpropagation



Backpropagation

- Treina a rede com pares entrada-saída
  - Cada vetor de entrada é associado a uma saída desejada
- Treinamento em duas fases, cada uma percorrendo a rede em um sentido
  - Fase forward
  - Fase backward
- Rumelhart, Hinton e Williams (1986)
  - Werbos (1974)

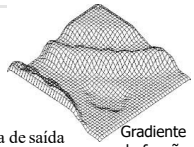


Backpropagation

- Treinamento
  - Supervisionado
  - Ajuste dos pesos: $\Delta w_{ij} = \eta x_i \delta_j$

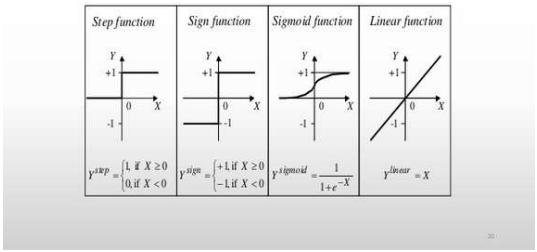
$$\delta_j = \begin{cases} f'(net) erro_j & \text{se j for camada de saída} \\ f'(net) \sum w_{jk} \delta_k & \text{se j for camada intemediária} \end{cases}$$

$$erro_j = \frac{1}{2} \sum_{q=1}^{c} (y_q - f(net_q)) \qquad net = \sum_{i=0}^{m} x_i w_i$$

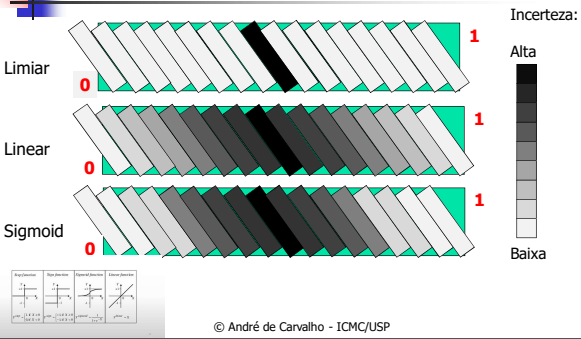


  - Se *f (net)* for uma função sigmoidal, *f´(net)=f(net)(1−f(net)*
  - Treinamento não é garantido de convergir

Funções de ativação



Funções de ativação e fronteiras



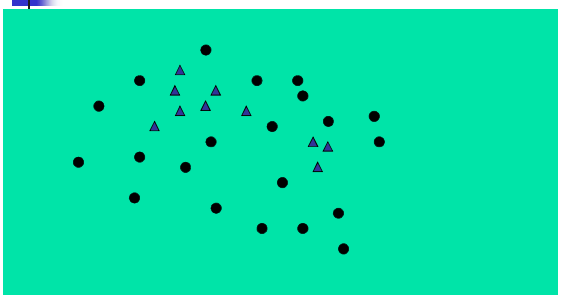
Funções de ativação

Name	Plot	Equation	Derivative
Sigmoid		$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
Tanh		$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$	$f'(x) = 1 - f(x)^2$
Rectified Linear Unit (relu)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Leaky Rectified Linear Unit (Leaky relu)		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$

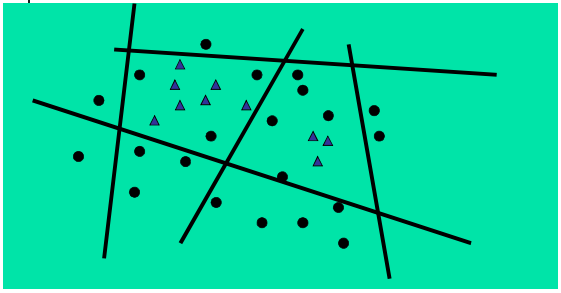
Algoritmo de treinamento

Iniciar todas as conexões com valores aleatórios ∈ [a,b]
Repita
erro = 0;
Para cada par de treinamento (X, y)
Para cada camada k := 1 a N
Para cada neurônio j := 1 a M_k
Calcular a saída f_{kj}(net)
Se k= N
Então Calcular soma dos erros dos neurônios da camada;
Se erro > ε
Então Para cada camada k := N a 1
Para cada neurônio j:= 1 a M_k
Atualizar pesos;
Até erro < ε (ou número máximo de ciclos)

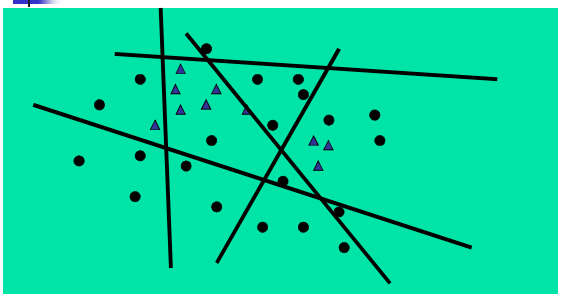
Treinamento modificando fronteiras



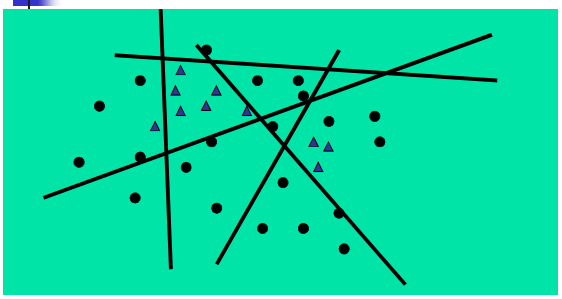
Treinamento modificando fronteiras



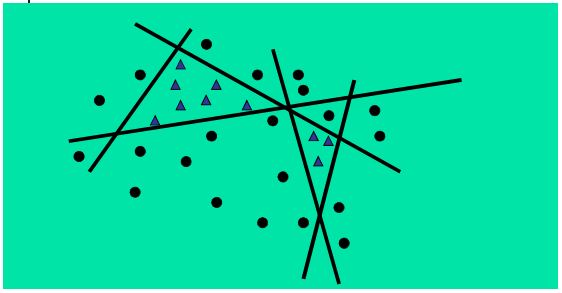
Treinamento modificando fronteiras



Treinamento modificando fronteiras



Treinamento modificando fronteiras

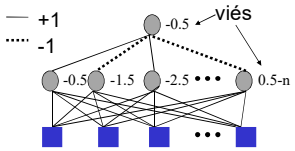


Exercício

Dada a rede abaixo, que recebe como entrada um vetor binário de *n* bits e gera como saída um valor binário:

a) Indicar a função implementada pela rede abaixo:
b) Explicar papel de cada neurônio no processamento da função

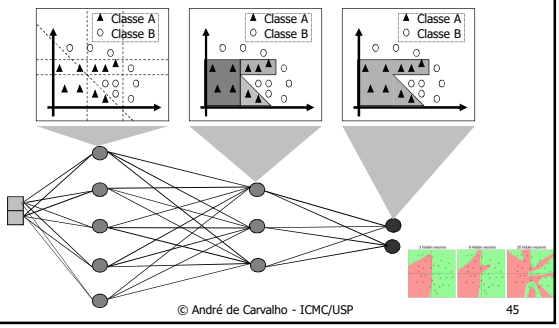
Considerar função de ativação limiar (threshold) entrada/saída binária



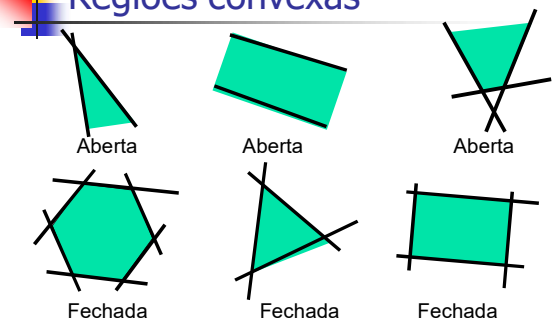
Exercício

- Paridade
  - Uma das limitações do Perceptron levantadas por Minsky e Papert
- Problema difícil
  - Padrões mais semelhantes requerem respostas diferentes
  - Usa *n* unidades intermediárias para detectar paridade em vetores com *n* bits

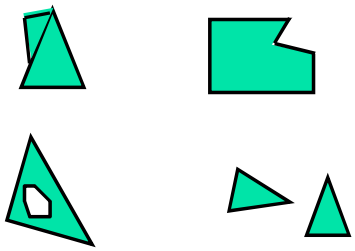
MLPs como classificadores



Regiões convexas

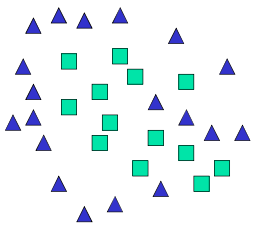


Combinações de regiões convexas



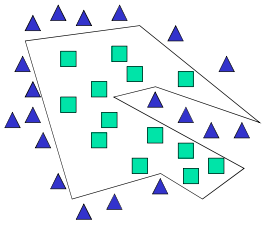
Combinações de regiões convexas

- Encontrar fronteiras de decisão que separem os dados abaixo:



Combinações de regiões convexas

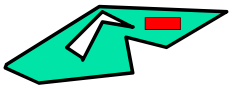
- Encontrar fronteiras de decisão que separem os dados abaixo:



Exercício

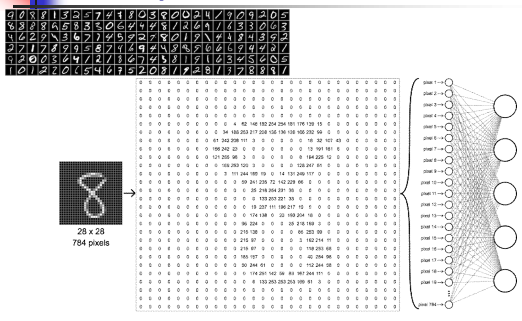
- Quantas camadas e pelo menos quantos neurônios em cada camada tem a rede que divide o espaço de entradas das formas abaixo:

classe 1
classe 2



classe 1
classe 2
classe 3

Exemplo



Conclusão

- Redes Neurais
  - Sistema nervoso
  - Muito utilizadas em problemas reais
    - Várias arquiteturas e algoritmos
  - Magia negra
  - Caixa preta

Perguntas?