Aprendizado de Máquina: Treinamento de Redes Neurais

Prof. Arnaldo Candido Junior
UTFPR – Medianeira

Treinamento Perceptron

- Modos de aprendizado
  - Modo online (**foco**): rede ajusta pesos a cada instância vista
  - Modo offline: rede ajusta pesos somente após analisar todas as instâncias
  - Modo batch (recomendado): ajustar pesos de **n** em **n** instâncias. Ex.: 100 em 100

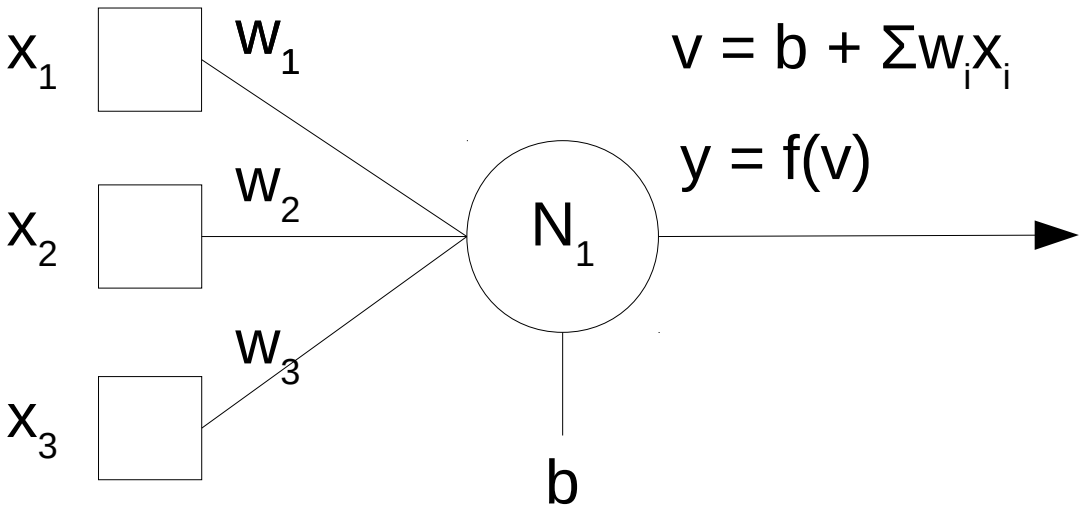
Treinamento Perceptron (2)

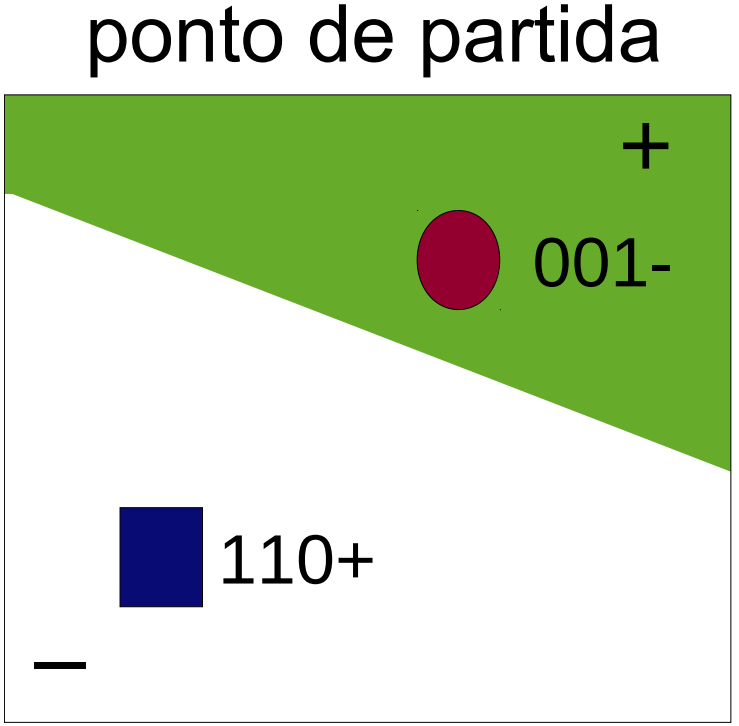
- Correção de erro no modo online:
  - $w_i = w_i - \eta x_i(y - d)$
  - $b = b - \eta(y - d)$
- Variação comum da mesa fórmula :
  - $w_i = w_i + \eta x_i(d - y)$
  - $b = b + \eta(d - y)$

Treinamento Perceptron (3)

- Pesos inicias: 0.4, -0.6, 0.6
- Bias inicial: -0.5
- Taxa de aprendizada (dada): 0.4

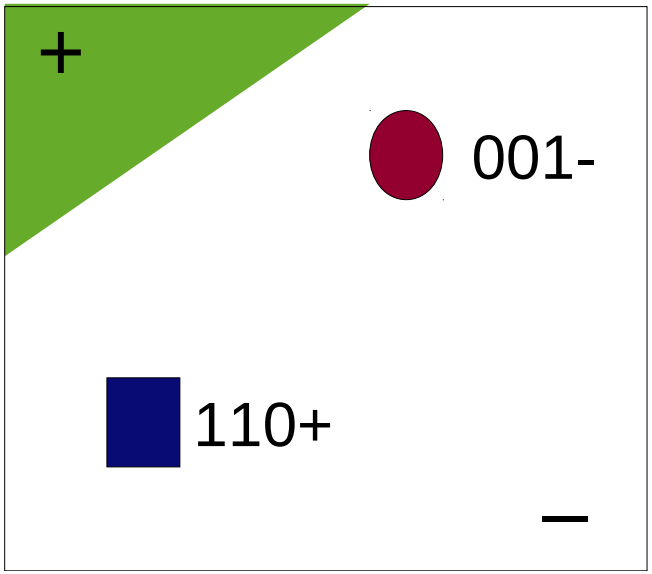
Treinamento Perceptron (3)





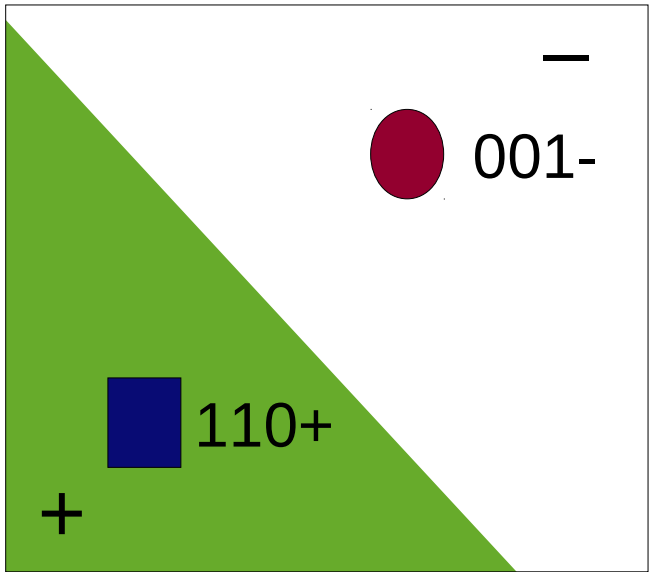
Treinamento Perceptron (4)

- Passo (a): padrão 001 (d = 0)
  v = 0*0.4 + 0*-0.6 + 1*0.6 - 0.5 = 0.1
  y = 1
- d ≠ y: atualizar pesos
  w_0 = 0.4 - 0.4(0)(1 - 0) = 0.4
  w_1 = -0.6 - 0.4(0)(1 - 0) = -0.6
  w_2 = 0.6 - 0.4(1)(1 - 0) = 0.2
  b = -0.5 - 0.4(1 - 0) = -0.9
- Exercício: padrão 110 (d = 1)



Treinamento Perceptron (5)

- Passo (b): padrão 110 (d = 1)
  v = 1(0.4) + 1(-0.6) + 0(0.2) - 0.9 = -1.1
  y = 0
- d ≠ y: atualizar pesos
  w_0 = 0.4 – 0.4(1)(0 - 1) = 0.8
  w_1 = -0.6 - 0.4(1)(0 - 1) = -0.2
  w_2 = 0.2 - 0.4(0)(0 - 1) = 0.2
  b = -0.9 - 0.4(0 - 1) = -0.5
- Exercício: padrão 001 (d = 0)



Treinamento Perceptron (6)

- Passo (c): padrão 001 (d = 0)
  v = 0*0.8 + 0*-0.2 + 1*0.2 - 0.5 = -0.3
  y = 0
  d = y: os pesos não precisam ser modificados
- Passo (d): padrão 110 (d = 1)
  v = 1(0.8) + 1(-0.2) + 0(0.2) - 0.5 = +0.1
  y = 1
  d = y: os pesos não precisam ser modificados

Treinamento Perceptron (7)

- Generalização de novas instâncias
  - Padrão 111
    v = 1*0.8 + 1*-0.2 + 1*0.2 - 0.5 = 0.3
    y = 1 ⇒ **classe +**
  - Padrão 000
    v = 0*0.8 + 0*-0.2 + 0*0.2) - 0.5 = -0.5
    y = 0 ⇒ **classe -**

Treinamento Perceptron (8)

- Generalização de novas instâncias
  - Padrão 100: exercício
  - Padrão 011: exercício
- Obs: a rede convergiu em apenas um ciclo de atualização de pesos (uma época)
  - Redes reais demoram mais para convergir

Treinamento MLP

- Dois algoritmos
  - Backpropagation: cálcula a direção de descida da encosta
  - Gradiente Descendente (ou similar): desce a encosta
- Necessário o uso função de custo diferenciável e uma função de ativação também diferenciável
  - Minimizar custo equivale a minimizar taxa de erro

Treinamento MLP (2)

- Exemplo a seguir
  - Rede Feedforward
  - Completamente Conectada
  - Baseada em ativação sigmoide
  - Com função de custo erros quadráticos médios
  - Modo de aprendizado online

Treinamento MLP (3)

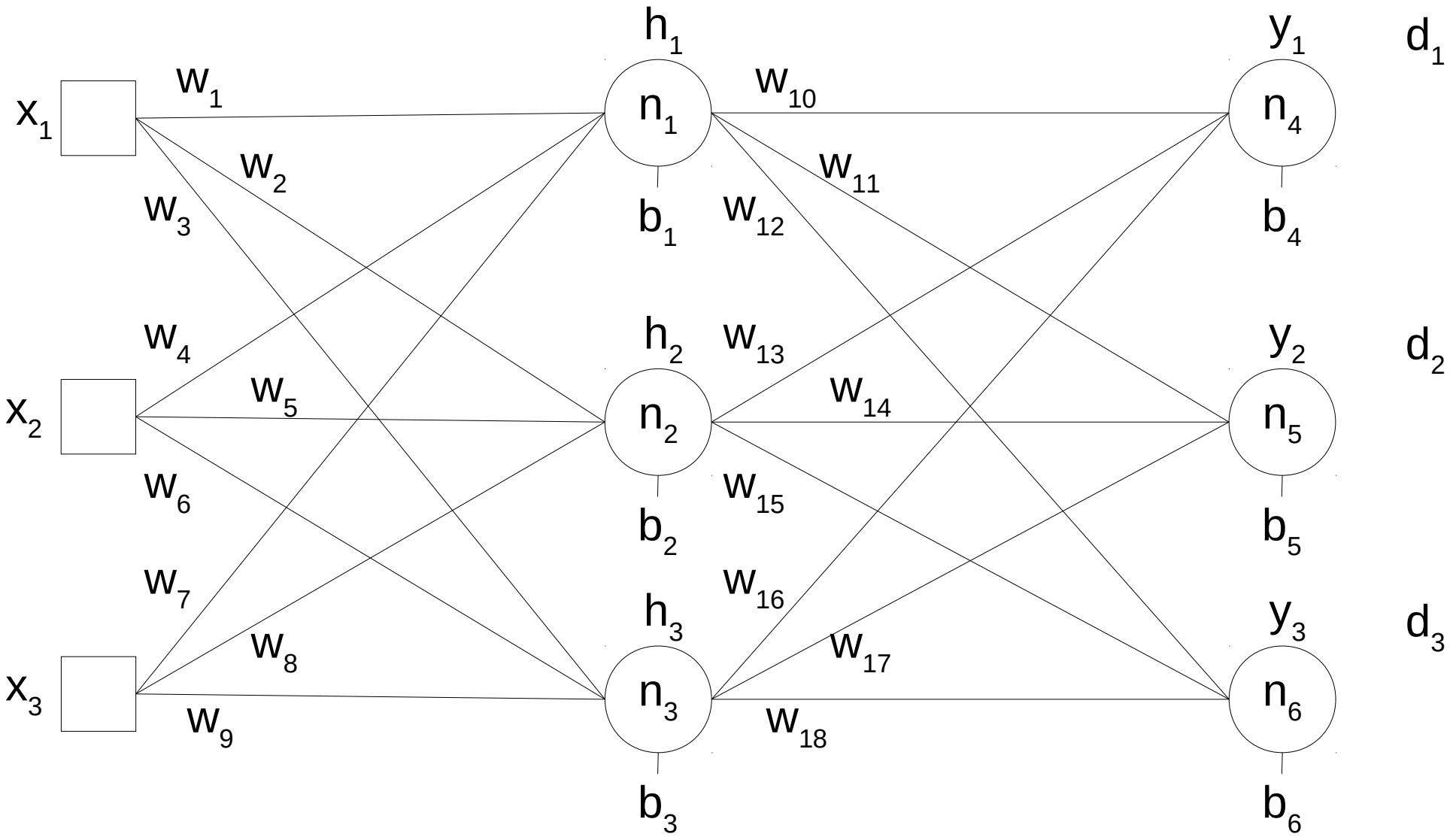
$w_{ij} = w_{ij} - \eta x_i \delta_j$

$b_j = b_j - \eta \delta_j$

$\delta_j = y_j(1 - y_j)(y_j - d_j)$ (última camada)

$\delta_j = y_j(1 - y_j)\sum w_{jk}\delta_k$ (camadas intermediárias)

Treinamento MLP (4)



Treinamento MLP (5)

- Fase forward na camada escondida: ativação h_3

  $v_3 = w_3x_1 + w_6x_2 + w_9x_3 + b_3$

  $h_3 = \sigma(v_3)$

- Fase forward na camada de saída: ativação y_1

  $v_4 = w_{10}h_1 + w_{13}h_2 + w_{16}h_3 + b_4$

  $y_1 = \sigma(v_4)$

- Demais ativações são análogas

Treinamento MLP (6)

- Fase backward na camada de saída:
  ajuste w_{10} e b_4

  $w_{10} = w_{10} - \eta h_1 \delta_4$
  $b_4 = b_4 - \eta \delta_4$
  $\delta_4 = y_1(1 - y_1)(y_1 - d_1)$
- Exercício: ajuste w_{17} e b_5 mostrando ativações envolvidas

Treinamento MLP (7)

- Fase backward na camada escondida: ajuste w_6 e b_3

  $w_6 = w_6 - \eta x_2 \delta_3$
  $b_3 = b_3 - \eta \delta_3$
  $\delta_3 = h_3(1 - h_3)(w_{16}\delta_4 + w_{17}\delta_5 + w_{18}\delta_6)$

- Exercício: ajuste w_2 e b_1 mostrando ativações envolvidas

Agradecimentos/referências

- Parcialmente adaptado de:
  Notas de aula do Prof. André de Carvalho (USP)