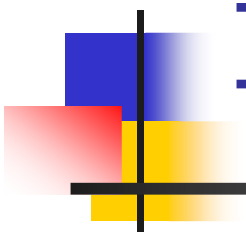
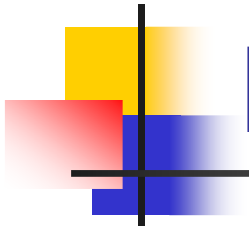


Inteligência Artificial

Prof. Tiago A. E. Ferreira

Aula 19 - Perceptron



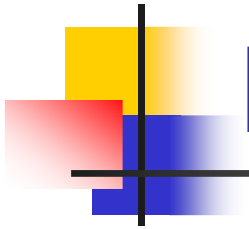
Perceptron

- Desenvolvido por Rosemblat (1958)
- Rede mais simples para a classificação de padrões linearmente separáveis

Perceptron

- Estado de ativação
  - 1 » Ativo
  - -1 » Inativo
- Função de ativação
  - $a_i(t+1) = \mathbf{f}(u_i(t))$
  - $a_i(t+1) =$
    - -1 se $u_i(t) < 0$
    - +1 se $u_i(t) \geq 0$

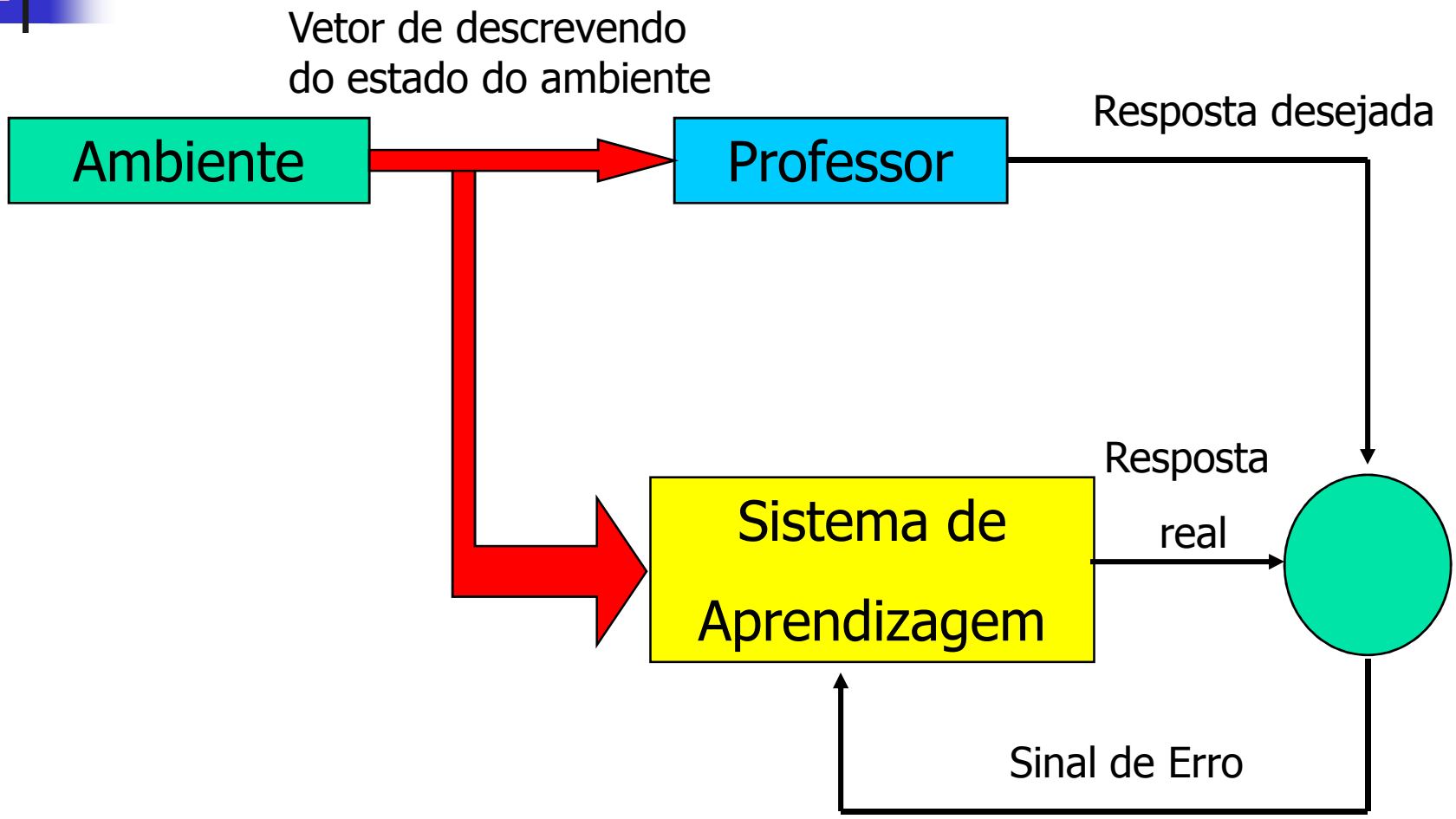
$$u = \sum_{j=1}^{N} x_j w_j - \theta$$



Perceptron

- Função de saída = função identidade
- Treinamento
  - Supervisionado
  - Correção de erro
    - $\Delta w_{ij} = \eta x_i(d_j - y_j)$ $(d \neq y)$
    - $\Delta w_{ij} = 0$ $(d = y)$
      - Onde a saída do perceptron é y e a saída desejada é d
- Teorema de Convergência: se é possível classificar um conjunto de entradas, uma rede Perceptron fará a classificação.

Aprendizado Supervisionado



Algoritmo de Treinamento

1) Iniciar todas as conexões com $w_{ij} = 0$;
2) Repita

    Para cada par de treinamento (X, d)

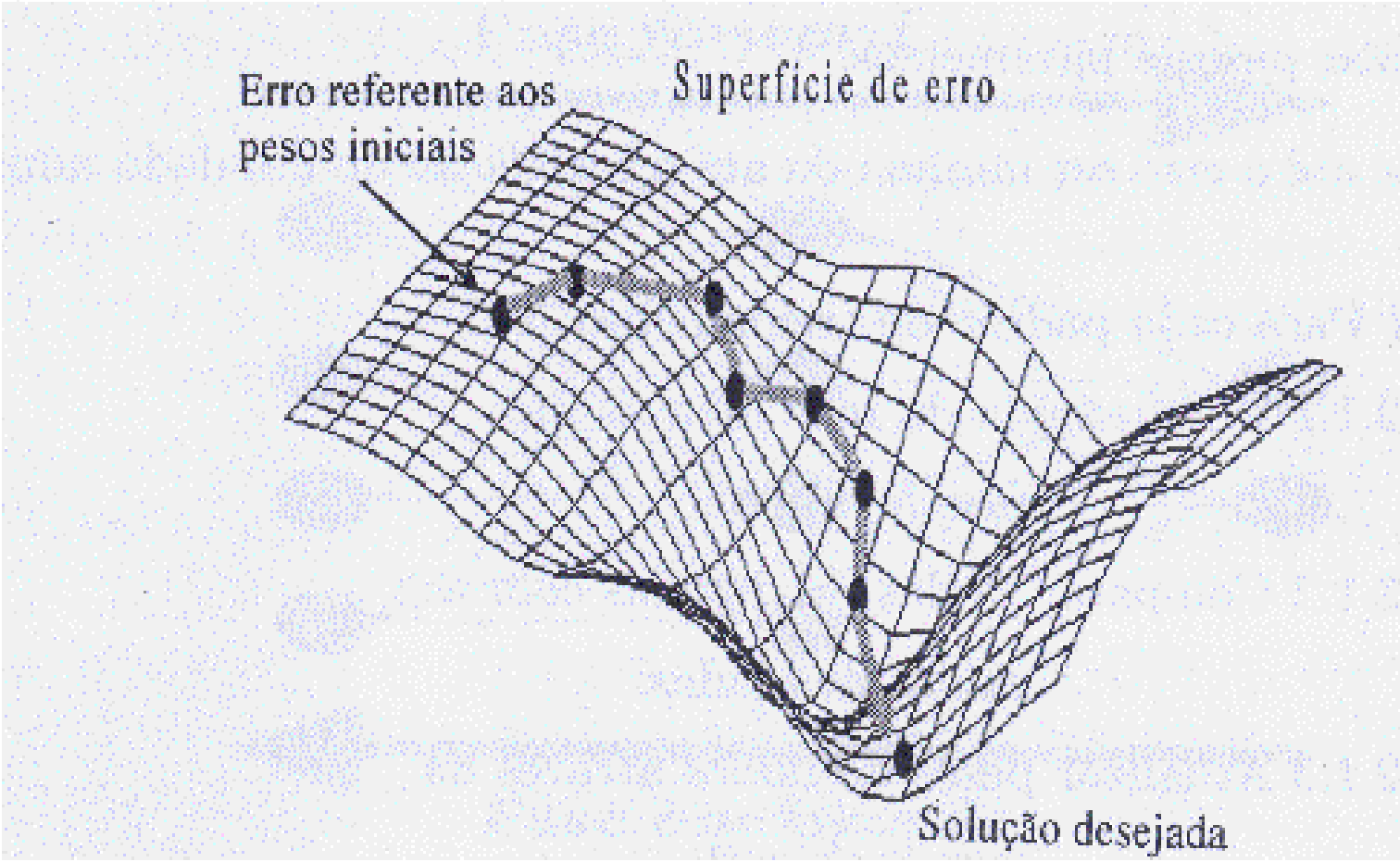
        Calcular a saída y

        Se (d ≠ y) então

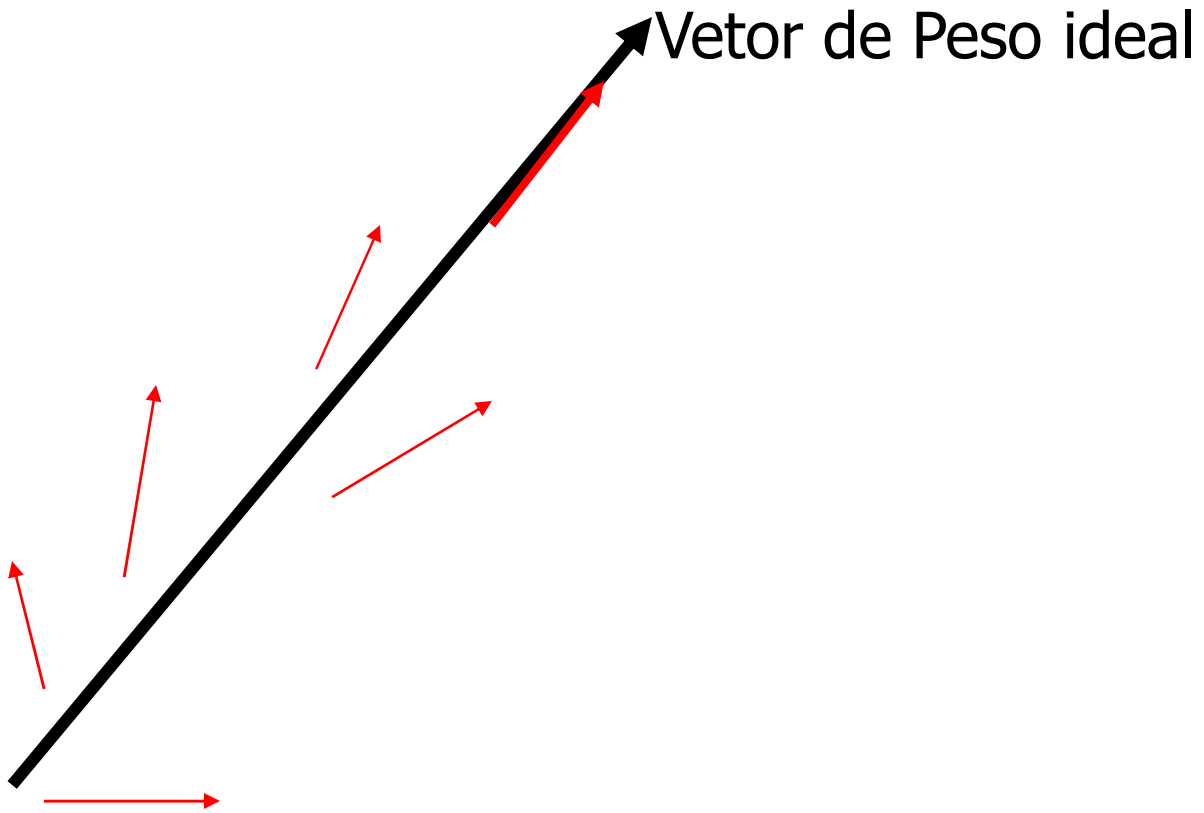
            Atualizar pesos dos neurônios

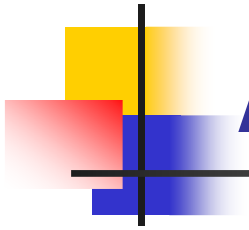
Até erro ser aceitável

Treinamento



Treinamento





Algoritmo de Teste

1) Apresentar padrão X a ser reconhecido
2) Calcular a saída y
3) Se (d = y)

    Então

        X∈ Classe 1

    Senão

        X∈ Classe 2

Exemplo

- Dada uma rede do tipo Percepetron formada por um neurônio com três entradas, que utiliza os pesos iniciais: W_0 = 0.4; W_1 = -0.6 e W_2 = 0.6 e limiar de 0.5 e uma taxa de aprendizado de 0.4. Responda os itens abaixo:
  a) Treinar a rede a gerar saída –1 para o padrão 001 e a saída +1 para padrão 110
  b) A que classe pertencem os padrões 111, 000, 100 e 011?

Exemplo1 – Item A

- Treinar a rede

1) Para o padrão 001 (d = -1)

Passo 1: definir saída da rede

$u=0(0.4)+0(-0.6)+1(0.6)-1(0.5)=0.1$

$y = u = +1$ (uma vez que $0.1 \geq 0$)

Passo 2: Atualizar os pesos

$W_0 = 0.4+0.4(0)(-1-(+1)) = 0.4$

$W_1 = -0.6+0.4(0)(-1-(+1)) = -0.6$

$W_2 = 0.6+0.4(1)(-1-(+1)) = -0.2$

$W_3 = 0.5+0.4(-1)(-1-(+1)) = 1.3$

Exemplo1 – Item A

- Treinar a rede

  2) Para o padrão 110 (d = 1)

  Passo 1: definir saída da rede

  u=1(0.4)+1(-0.6)+0(-0.2)-1(1.3)=-1.5

  y = u = -1 (uma vez que –1.5 < 0)

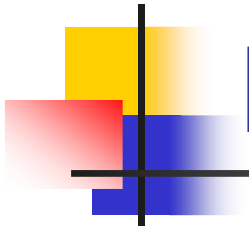
  Passo 2: Atualizar os pesos

  W_0 = 0.4+0.4(1)(1-(-1)) = 1.2

  W_1 = -0.6+0.4(1)(1-(-1)) = 0.2

  W_2 = -0.2+0.4(0)(1-(-1)) = -0.2

  W_3 = 1.3+0.4(-1)(1-(-1)) = 0.5



Exemplo1 – Item A

- Treinar a rede

  3) Para o padrão 001 (d = -1)

  Passo 1: definir saída da rede

  u=0(1.2)+0(0.2)+1(-0.2)-1(0.5)=-0.7

  y = u = -1 (uma vez que –0.7 < 0)

  Passo 2: Atualizar os pesos

  Como y = d, os pesos não precisam ser modificados

Exemplo1 – Item A

- Treinar a rede

  4) Para o padrão 110 (d = 1)

  Passo 1: definir saída da rede

  $u=1(1.2)+1(0.2)+0(-0.2)-1(0.5)=0.9$

  $y = u = 1$ (uma vez que $0.9 \geq 0$)

  Passo 2: Atualizar os pesos

  Como y = d, os pesos não precisam ser modificados

Exemplo 1 – item b

- Testar a rede

  1) Para o padrão 111

  $u = 1(1.2)+1(0.2)+1(-0.2)-1(0.5)=0.7$

  $y = u = 1$ (pois $0.7 \geq 0$) » Classe 1

  2) Para o padrão 000

  $u = 0(1.2)+0(0.2)+0(-0.2)-1(0.5)=-0.5$

  $y = u = -1$ (pois $-0.5 < 0$) » Classe 0

Exemplo 1 – item b

- Testar a rede

  3) Para o padrão 100

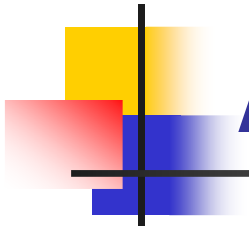
  $u = 1(1.2)+0(0.2)+0(-0.2)-1(0.5)=0.7$

  $y = u = 1$ (pois $0.7 \geq 0$) » Classe 1

  2) Para o padrão 011

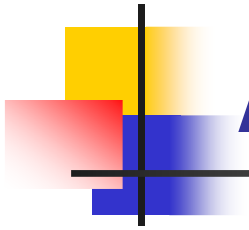
  $u = 0(1.2)+1(0.2)+1(-0.2)-1(0.5)=-0.5$

  $y = u = -1$ (pois $-0.5 < 0$) » Classe 0



Adaline

- Problema do Perceptron:
  - Ajuste de pesos não leva em conta a distância entre saída e resposta desejada
- Adaline proposto em 1960 por Widrow e Hoff



Adaline

- Estados de ativação
  - 1 » ativo
  - 0 » inativo
- Função de ativação
  - $a_i(t+1) = u_i(t)$
- Função de saída = função identidade

Adaline

- Treinamento
  - Supervisionado
  - Correção de erro (regra LMS – delta, Widrow-Holf)
    - $\Delta w_{ij} = \eta x_i(d_j - y_j)$ $(d \neq y)$
    - $\Delta w_{ij} = 0$ $(d = y)$
    - Reajuste gradual dos pesos
      - Leva em conta a distância entre a saída e a resposta desejada

Algoritmo de treinamento

1) Iniciar todas as conexões com $w_{ij} = 0$;
2) Repita

Para cada par de treinamento (X, d)

Calcular a saída y

Se (d – y > erro aceitável) então

Atualizar pesos dos neurônios

Até erro ser aceitável

Algoritmo de Teste

1) Apresentar padrão X a ser reconhecido
2) Calcular a saída y
3) Se (y > limite_sup)

Então

X∈ Classe 0

Senão

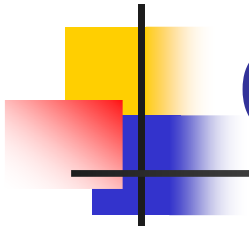
Se (y < limite_inf)

então

X∈ Classe 2

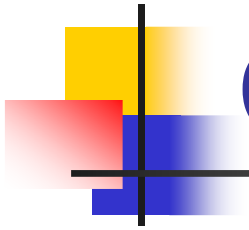
Senão

não é possível definir a classe de x



Conclusões

- Redes de uma única camada mapeiam entradas semelhantes em saídas semelhantes
  - Similaridade determinada pela *super-posição* dos padrões
  - Impedem o sistema de aprender certos mapeamentos
    - Estruturas de similaridades das entradas e das saídas são diferentes
      - Exemplos: paridade, ou-exclusivo



Conclusões

- Neste caso, a rede precisa criar uma representação interna das camadas
  - Necessário utilizar uma ou mais camdas intermediárias
  - Como treinar redes com mais de uma camada???????
    - *Backpropagaton!*