Perceptrons e Multilayer Perceptrons

Profa. Teresa Bernarda Ludermir

Aprendizagem de Máquina

Perceptrons

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

Perceptrons

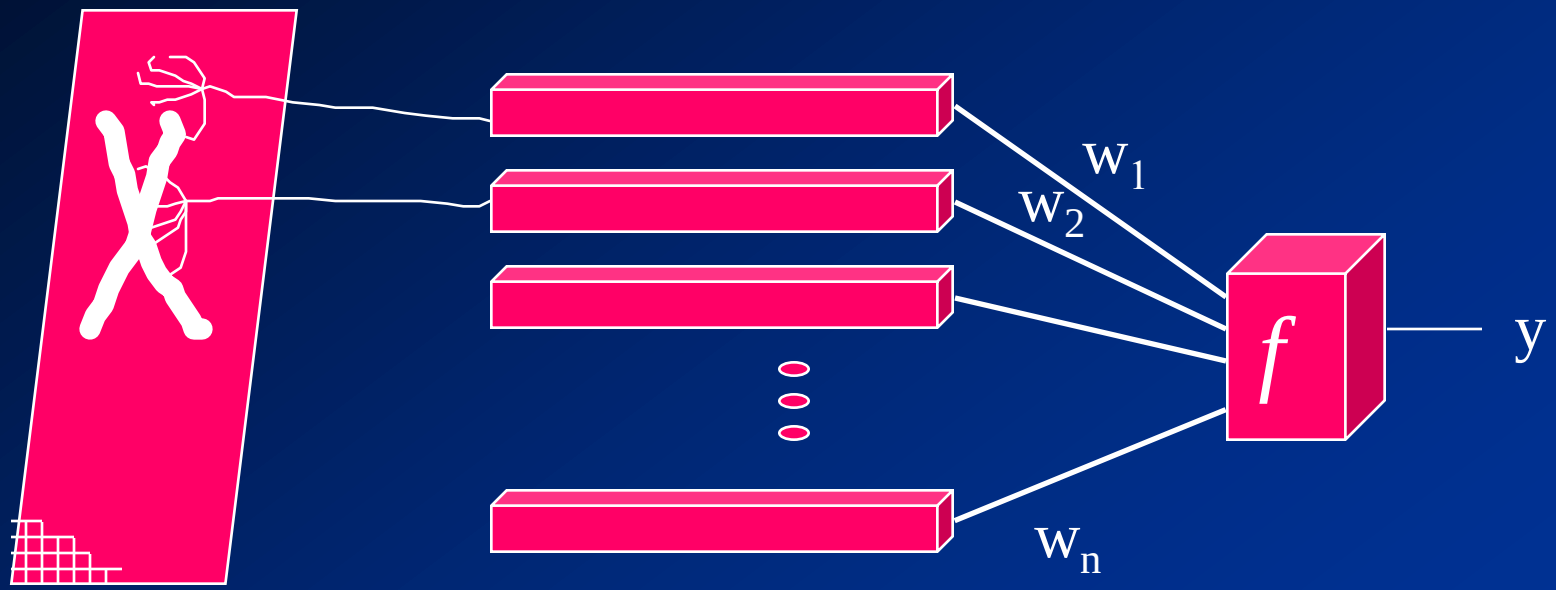
- Estado de ativação
  - 1 = ativo
  - -1 = inativo
- Função de ativação
  - $a_i(t+1) = f(u_i(t))$
  - $a_i(t+1) = \begin{cases} -1, & \text{se } u_i(t) < 0 \\ +1, & \text{se } u_i(t) \geq 0 \end{cases}$

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

Perceptrons

- Função de saída = função identidade
- Duas camadas
  - Camada de pré-processamento
    - M máscaras fixas para extração de características
      - Podem implementar qualquer função, mas pesos são fixos
  - Camada de discriminação
    - Uma unidade de saída para discriminar padrões de entrada
    - Pesos determinados através de aprendizado

Perceptrons



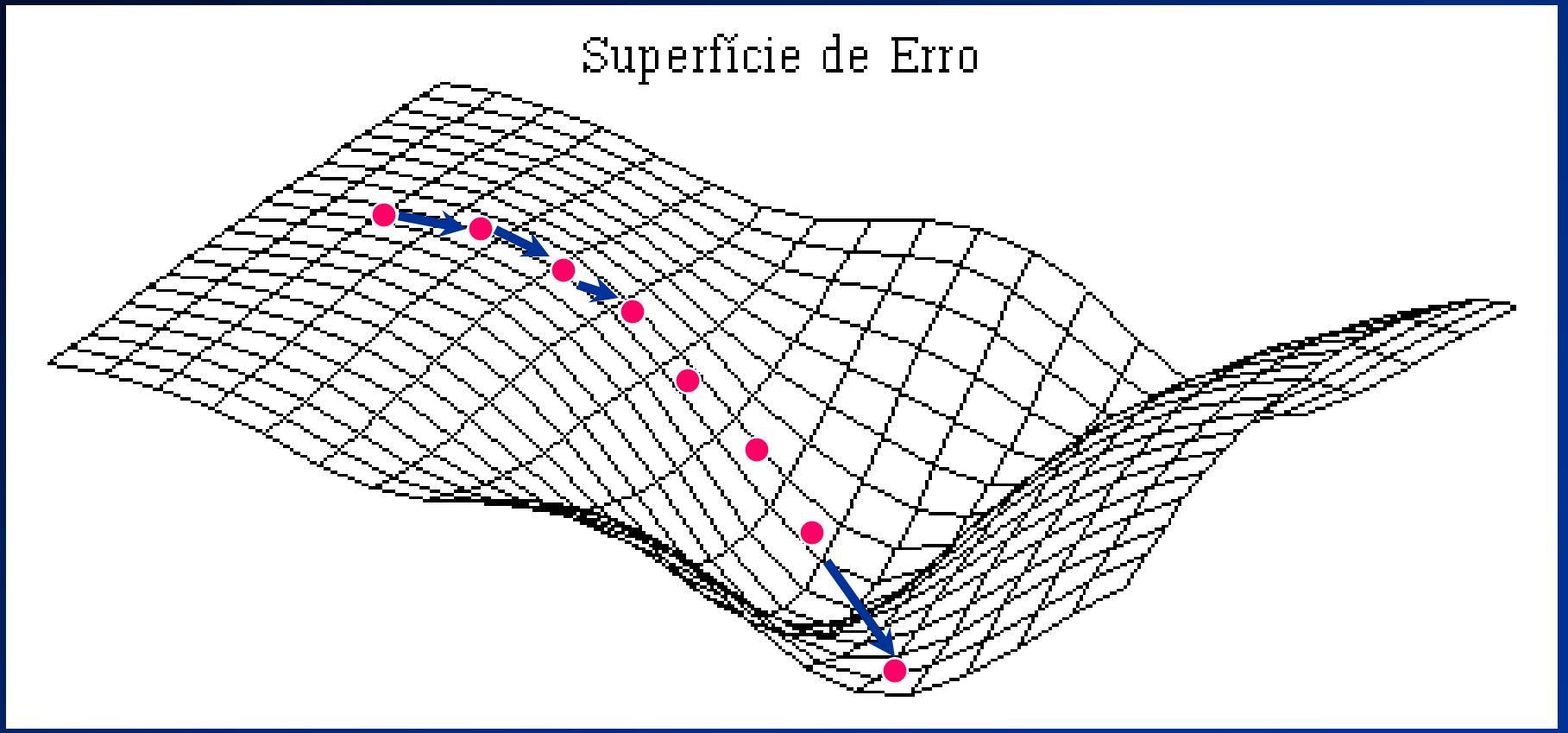
Perceptrons

- Treinamento
  - Supervisionado
  - Correção de erro
    - $\Delta w_{ij} = \eta x_i(d_j - y_j)$ $(d \neq y)$
    - $\Delta w_{ij} = 0$ $(d = y)$
- Teorema de convergência: se é possível classificar um conjunto de entradas, uma rede Perceptron fará a classificação

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 ($d \neq y$)
        Então
            Atualizar pesos dos neurônios
    Até o erro ser aceitável

Treinamento



Algoritmo de teste

1) Apresentar padrão X a ser reconhecido
2) Calcular a saída y
3) Se (y=-1)
   Então
      X ∈ classe 0
   Senão
      X ∈ classe 1

Exemplo

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 $w_2 = 0.6$, limiar $\theta = 0.5$ e uma taxa de aprendizado $= 0.4$, responda os itens abaixo:

a) Ensinar a rede a gerar a saída -1 para o padrão 001 e a saída +1 para os padrão 110

b) A que classe pertencem os padrões 111, 000, 100 e 011?

Exemplo 1: resposta a

a) Treinar a rede

a.1) Para o padrão 001 (d = -1)

Passo 1: definir a saída da rede

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

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

Passo 2: atualizar os pesos

$w_0 = 0.4 + 0.4(0)(-1 - (+1)) = 0.4$

$w_1 = -0.6 + 0.4(0)(-1 - (+1)) = -0.6$

$w_2 = 0.6 + 0.4(1)(-1 - (+1)) = -0.2$

$w_3 = 0.5 + 0.4(-1)(-1 - (+1)) = 1.3$

Exemplo 1: resposta a

a) Treinar a rede

a.2) Para o padrão 110 (d = 1)

Passo 1: definir a saída da rede

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

y = u = -1 (uma vez -1.5 < 0)

Passo 2: atualizar 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_2 = 1.3 + 0.4(-1)(1 - (-1)) = 0.5

Exemplo 1: resposta a

a) Treinar a rede

a.3) Para o padrão 001 (d = -1)

Passo 1: definir a saída da rede

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

y = u = -1 (uma vez -0.7 < 0)

Passo 2: atualizar pesos

Como d = y, os pesos não precisam ser modificados

Exemplo 1: resposta a

a) Treinar a rede

a.4) Para o padrão 110 (d = 1)

Passo 1: definir a saída da rede

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

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

Passo 2: atualizar pesos

Como d = y, os pesos não precisam ser

Exemplo 1: resposta b

b) Testar a rede

b.1) Para o padrão 111

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

$y = u = 1$ (porque $0.7 \geq 0$)) $\Rightarrow$ classe 1

b.2) Para o padrão 000

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

$y = u = -1$ (porque $-0.5 < 0$) $\Rightarrow$ classe 0

Exemplo 1: resposta b

b) Testar a rede

b.3) Para o padrão 100

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

$y = u = 1$ (porque $0.7 \geq 0$) $\Rightarrow$ classe 1

b.4) Para o padrão 011

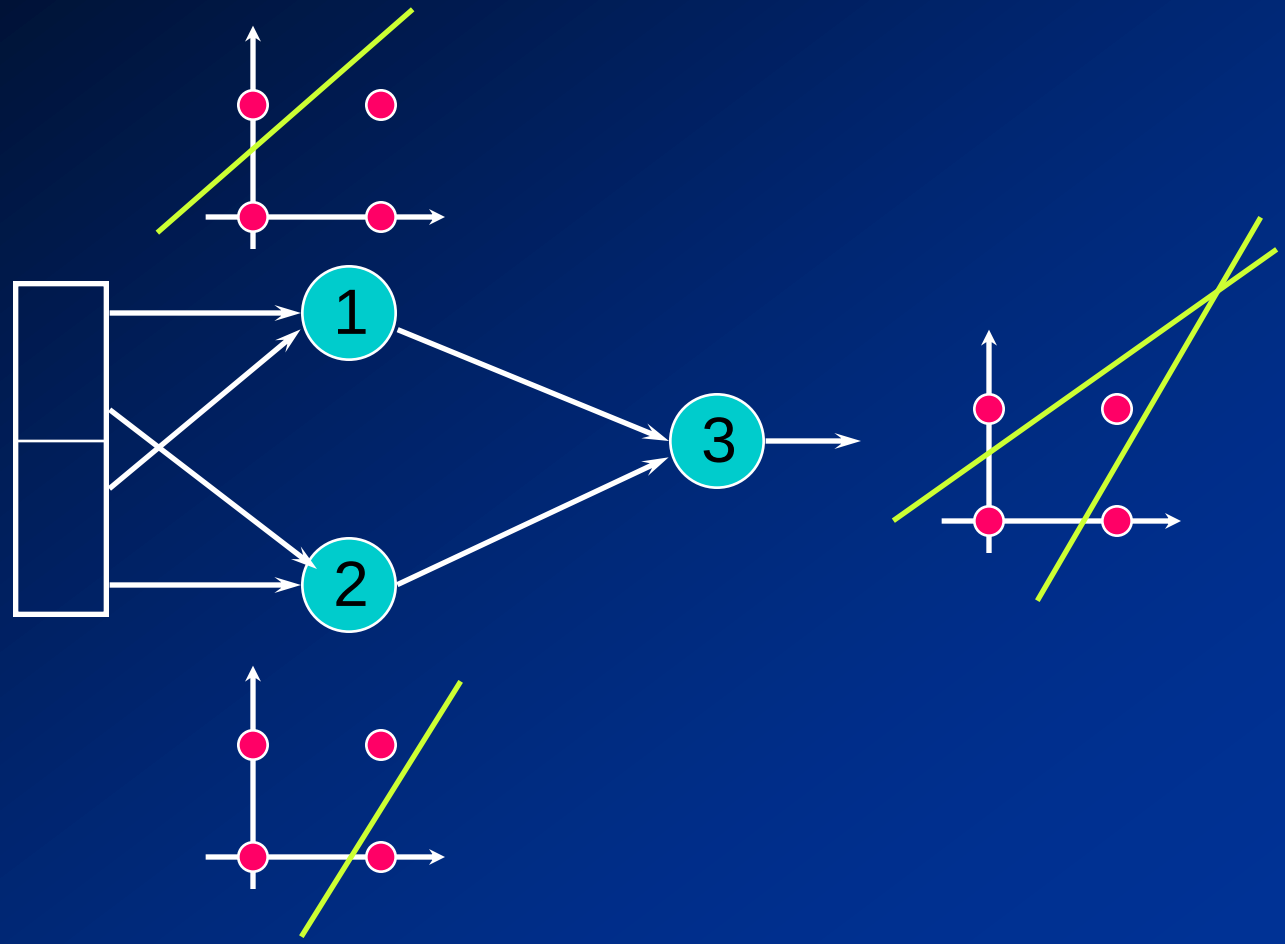
$u = 0(1.2) + 1(0.2) + 1(-0.2) - 1(0.5) = -0.5$

$y = u = -1$ (porque $-0.5 < 0$) $\Rightarrow$ classe 0

MLP - Introdução

- Redes de uma camada resolvem apenas problemas linearmente separáveis
- Solução: utilizar mais de uma camada
  - Camada 1: uma rede Perceptron para cada grupo de entradas linearmente separáveis
  - Camada 2: uma rede combina as saídas das redes da primeira camada, produzindo a classificação final

MLP - Introdução



MLP - Introdução

- Treinamento da rede
  - Treinar cada rede indepentemente
    - Saber como dividir o problema em sub-problemas
    - Nem sempre é possível
  - Treinar a rede toda
    - Qual o erro dos neurônios da camada intermediária?
    - Função threshold leva ao problema de atribuição de crédito
    - Usar função de ativação linear?

MLP - Introdução

- Função de ativação linear
  - Cada camada computa uma função linear
    - Composição de funções lineares é uma função linear
    - Sempre vai existir uma rede com uma camada equivalente uma rede multicamadas com funções de ativação lineares

MLP - Introdução

- Função de ativação para redes multicamadas
  - Não deve ser linear
  - Deve informar os erros para as camadas inferiores da rede
    - Função sigmóide
    - Função tangente hiperbólica

Rede Multi-Layer Perceptron

- Arquitetura de RNA mais utilizada
- Possuem uma ou mais camadas intermediárias de nós
  - Função de ativação mais utilizada é sigmóide logística

Treinamento de redes MLP

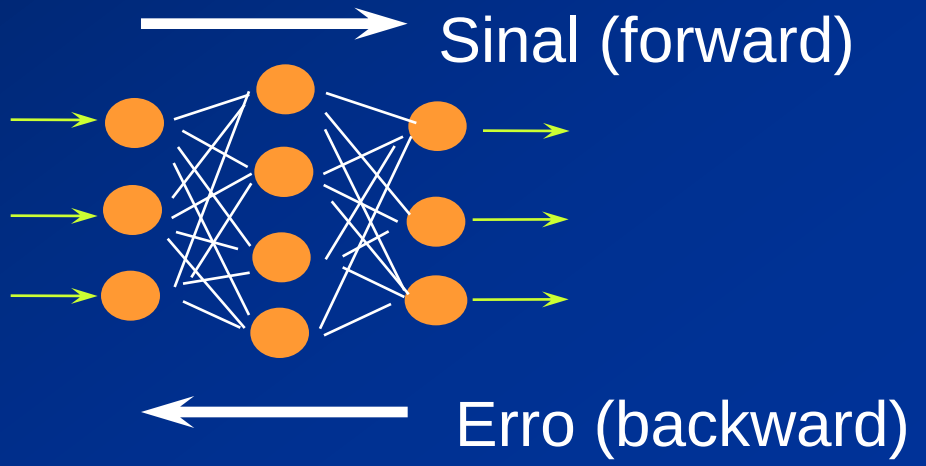
- Grande variedade de Algoritmos
  - Geralmente supervisionados
  - Estáticos
    - Não alteram estrutura da rede
    - Backpropagation, Função de Base Radial
  - Construtivos
    - Alteram estrutura da rede
    - Upstar, Cascade Correlation

Treinamento de redes MLP

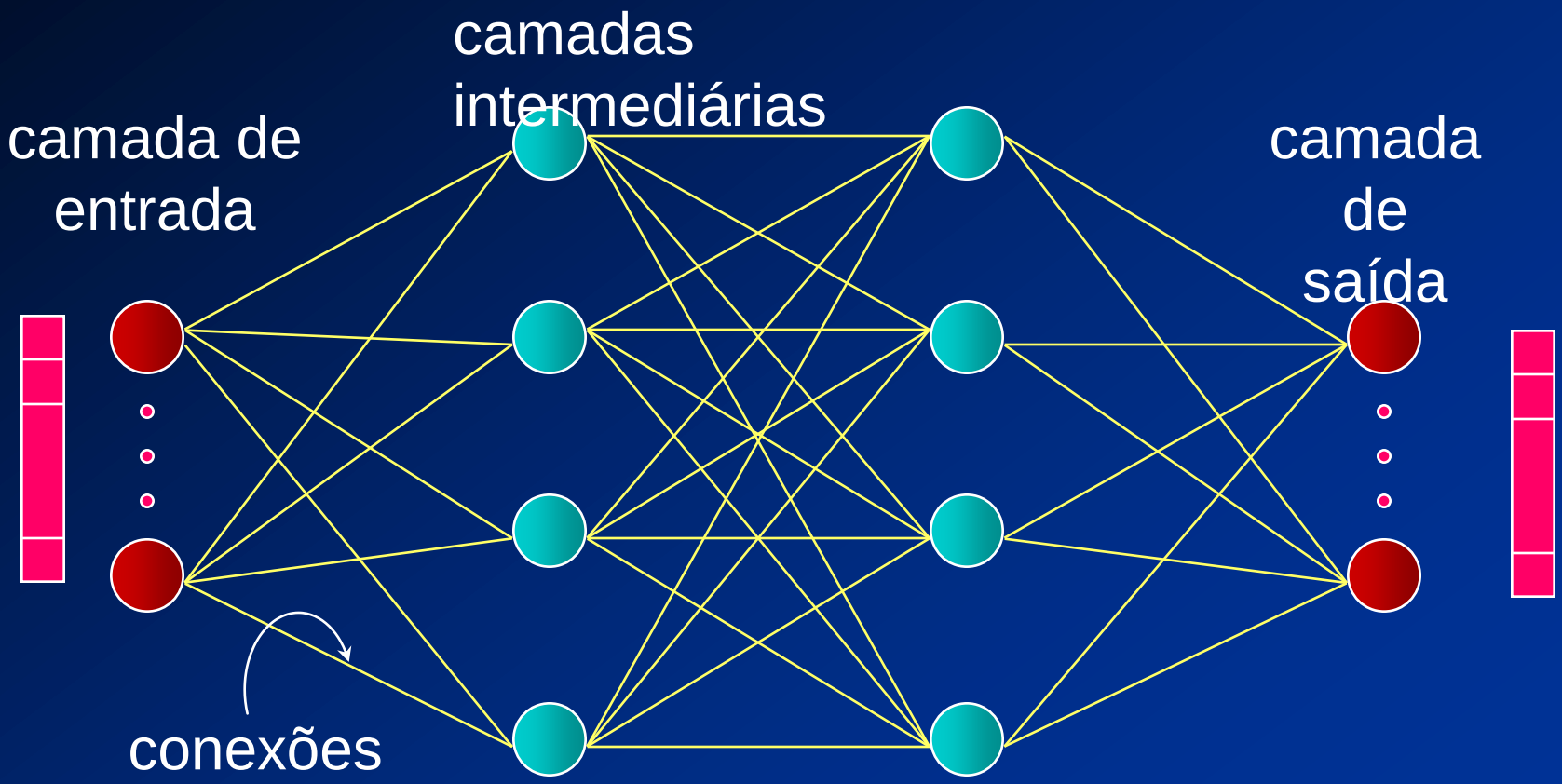
- Treinamento estático
  - MLPs com formatos e tamanhos diferentes podem utilizar mesma regra de aprendizado
    - Topologias diferentes podem resolver o mesmo problema
  - Regra mais utilizada: backpropagation

Backpropagation

- Rede é treinada com pares entrada-saída
- Cada entrada de treinamento está associada a uma saída desejada
- Treinamento em duas fases, cada uma percorrendo a rede em um sentido
  - Fase forward
  - Fase backward



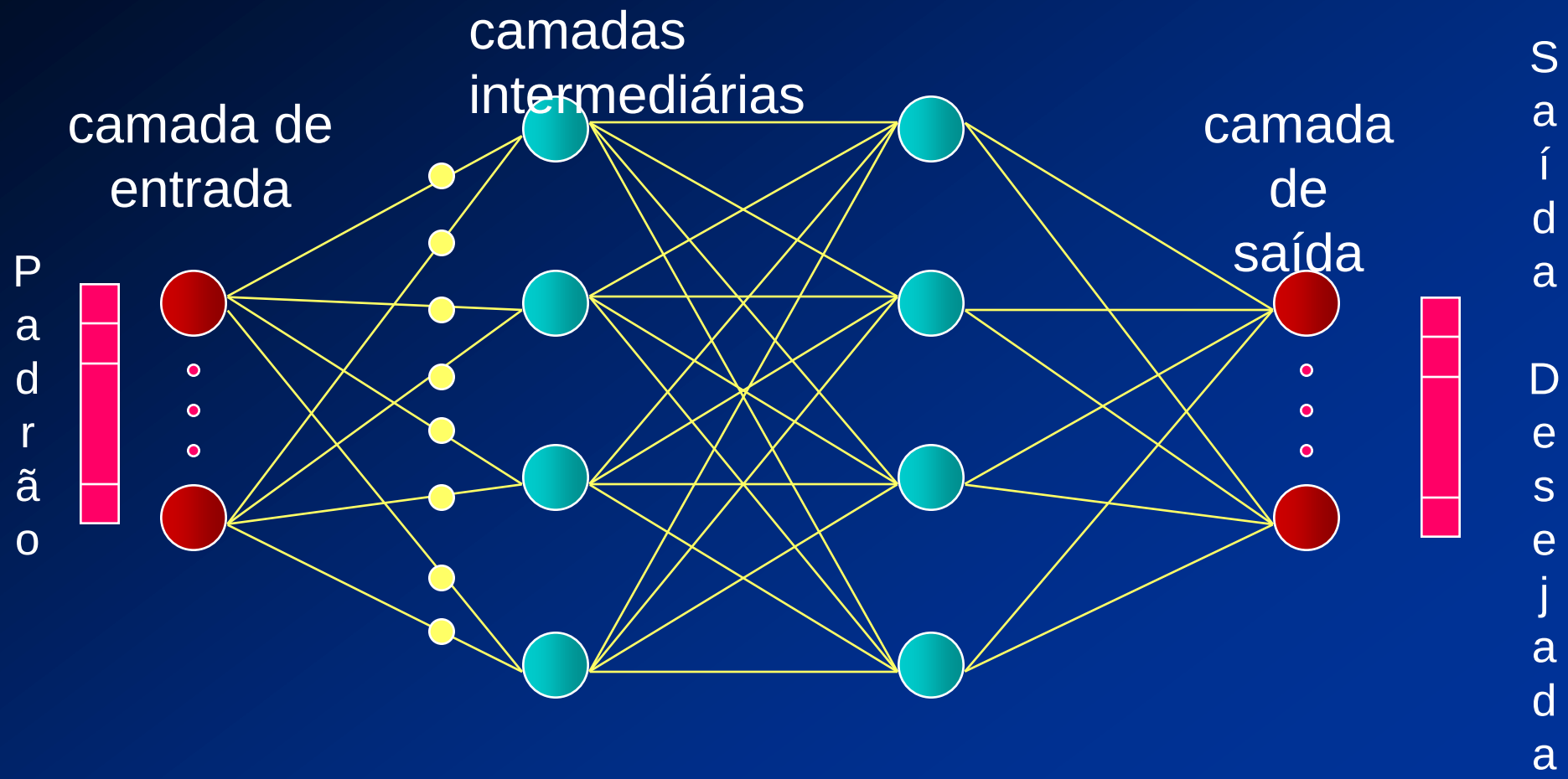
Rede MLP


camadas
intermediárias
camada de
entrada
camada
de
saída
conexões

Aprendizado

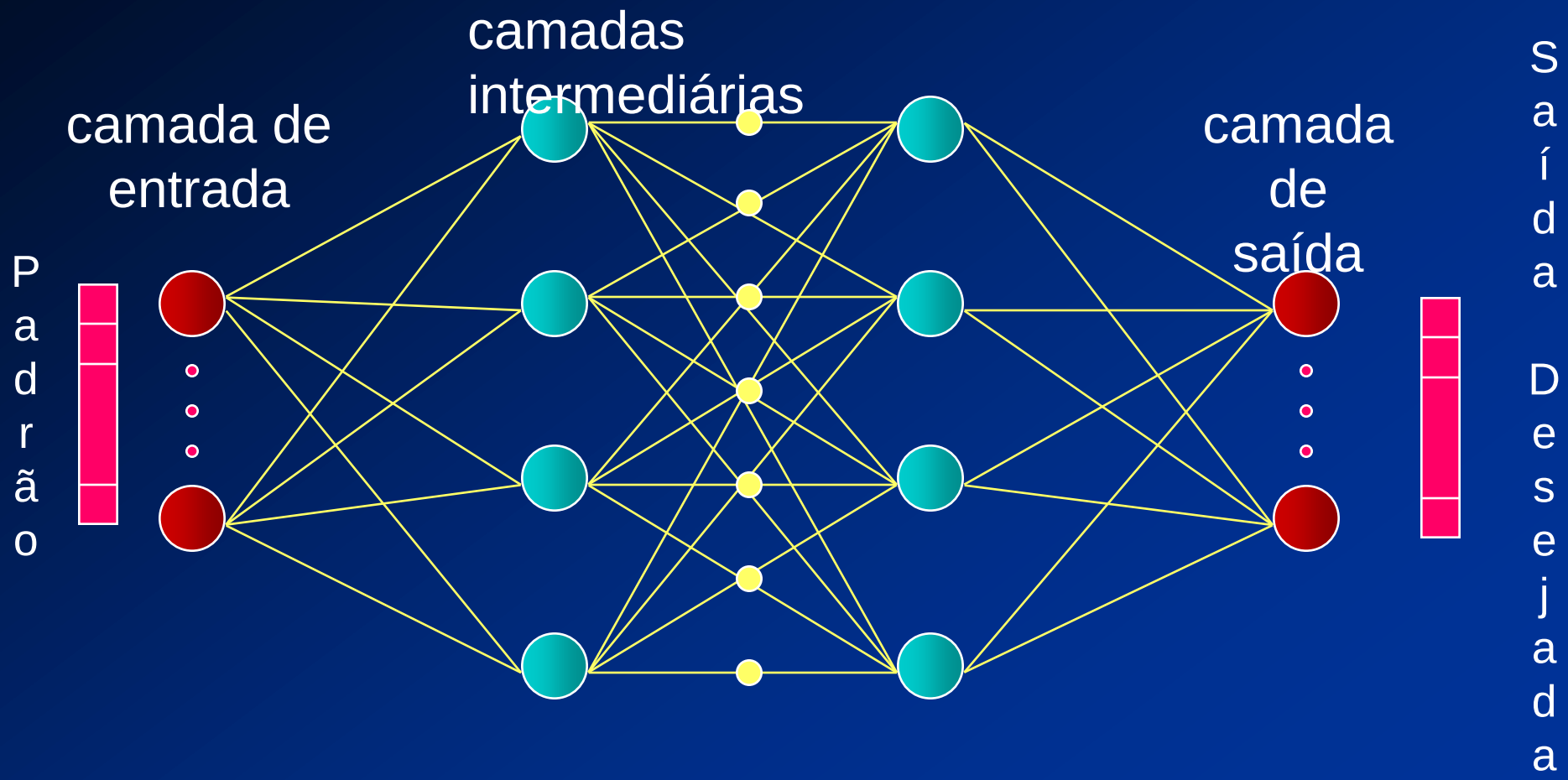
RNA - Aprendizado

RNA - Aprendizado



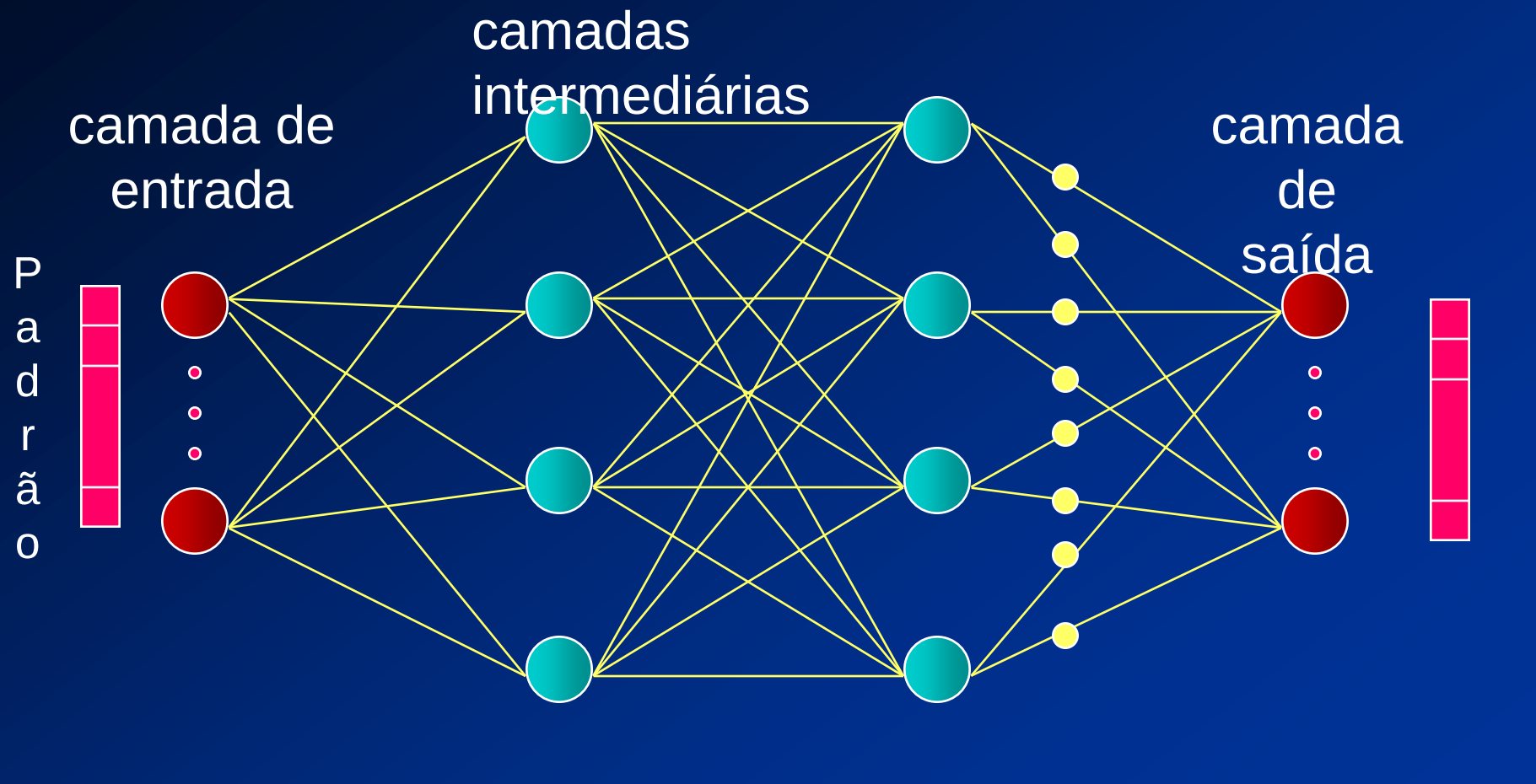
RNA - Aprendizado

RNA - Aprendizado



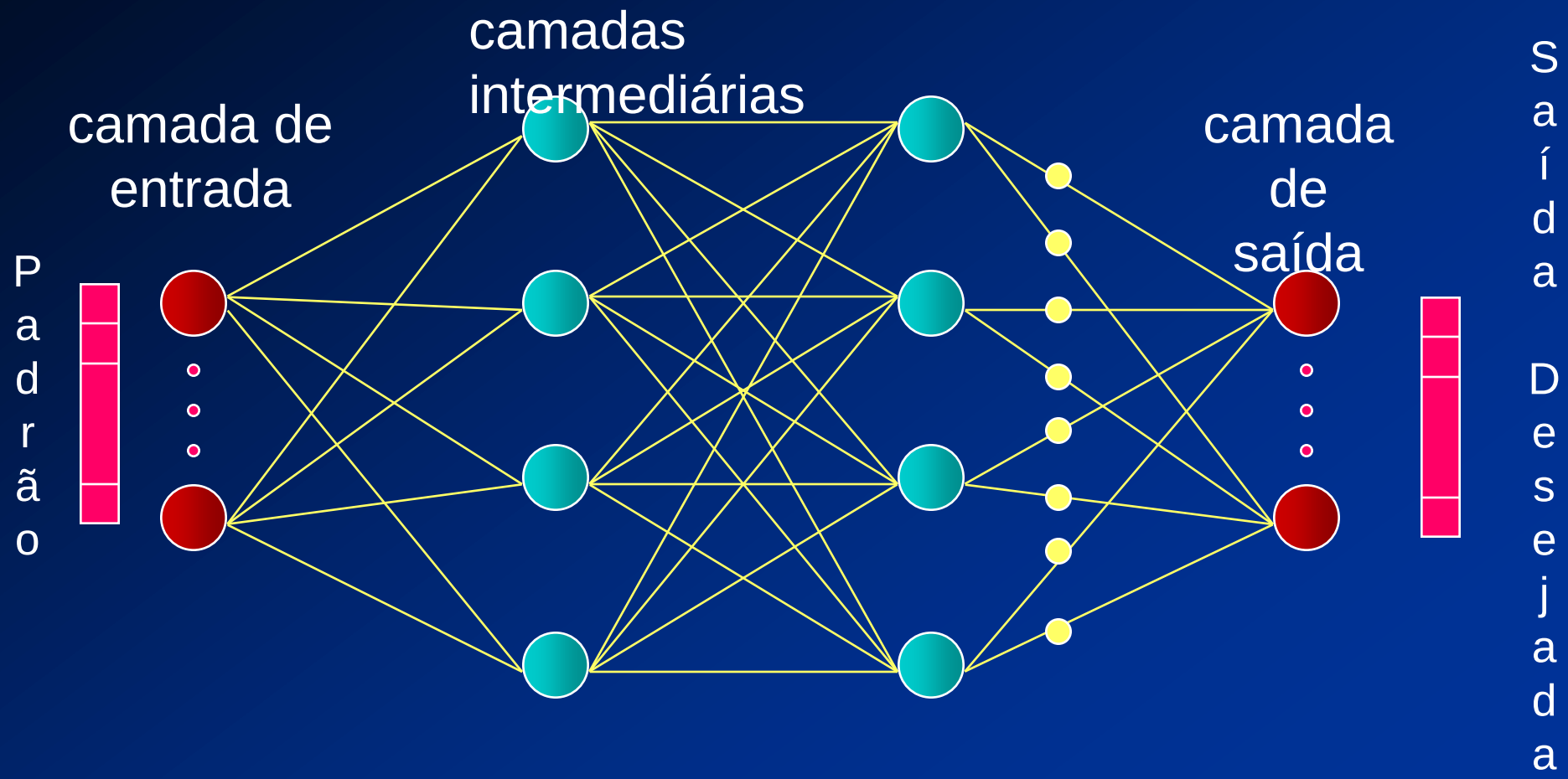
RNA - Aprendizado

RNA - Aprendizado

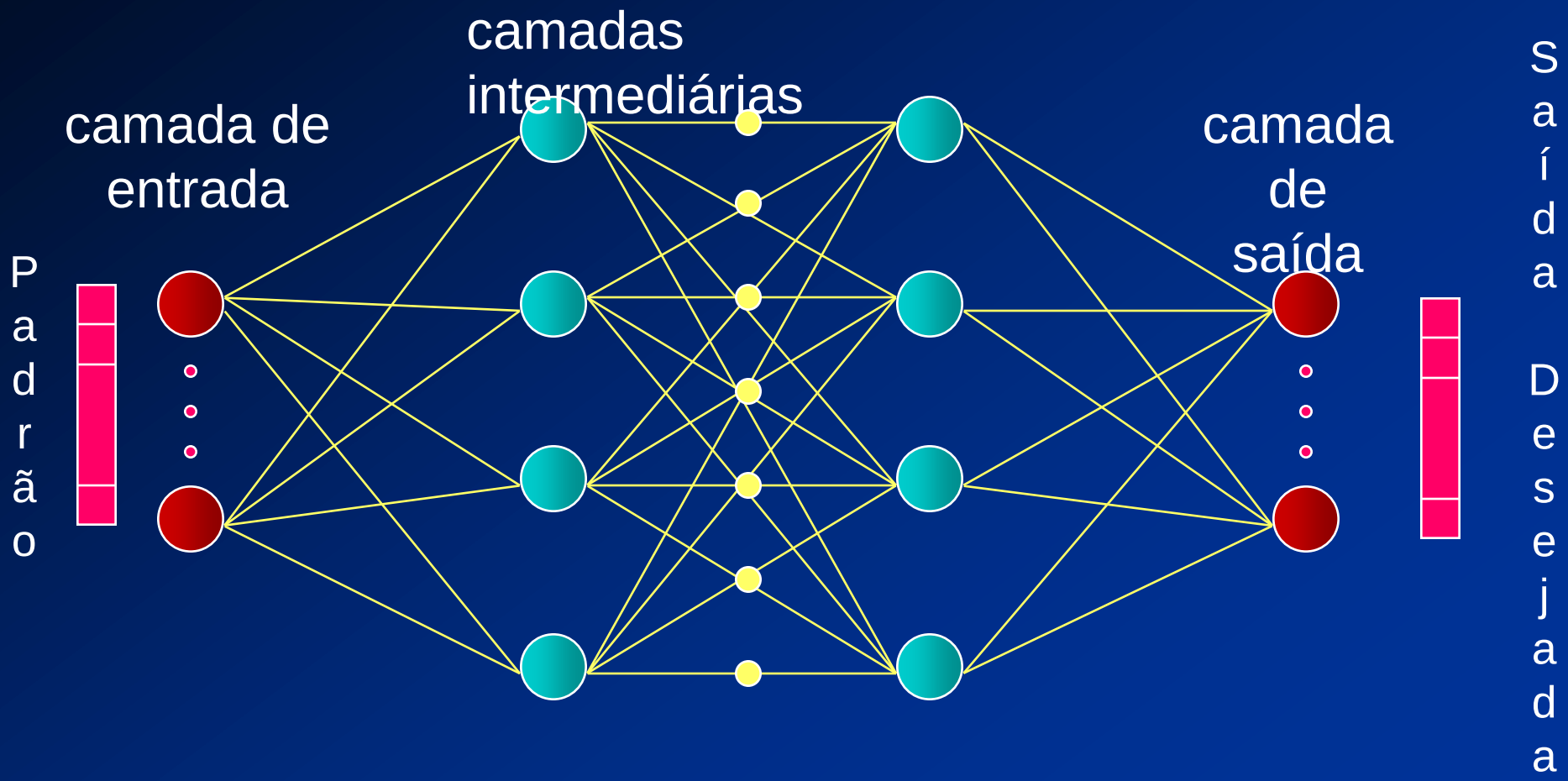


RNA - Aprendizado

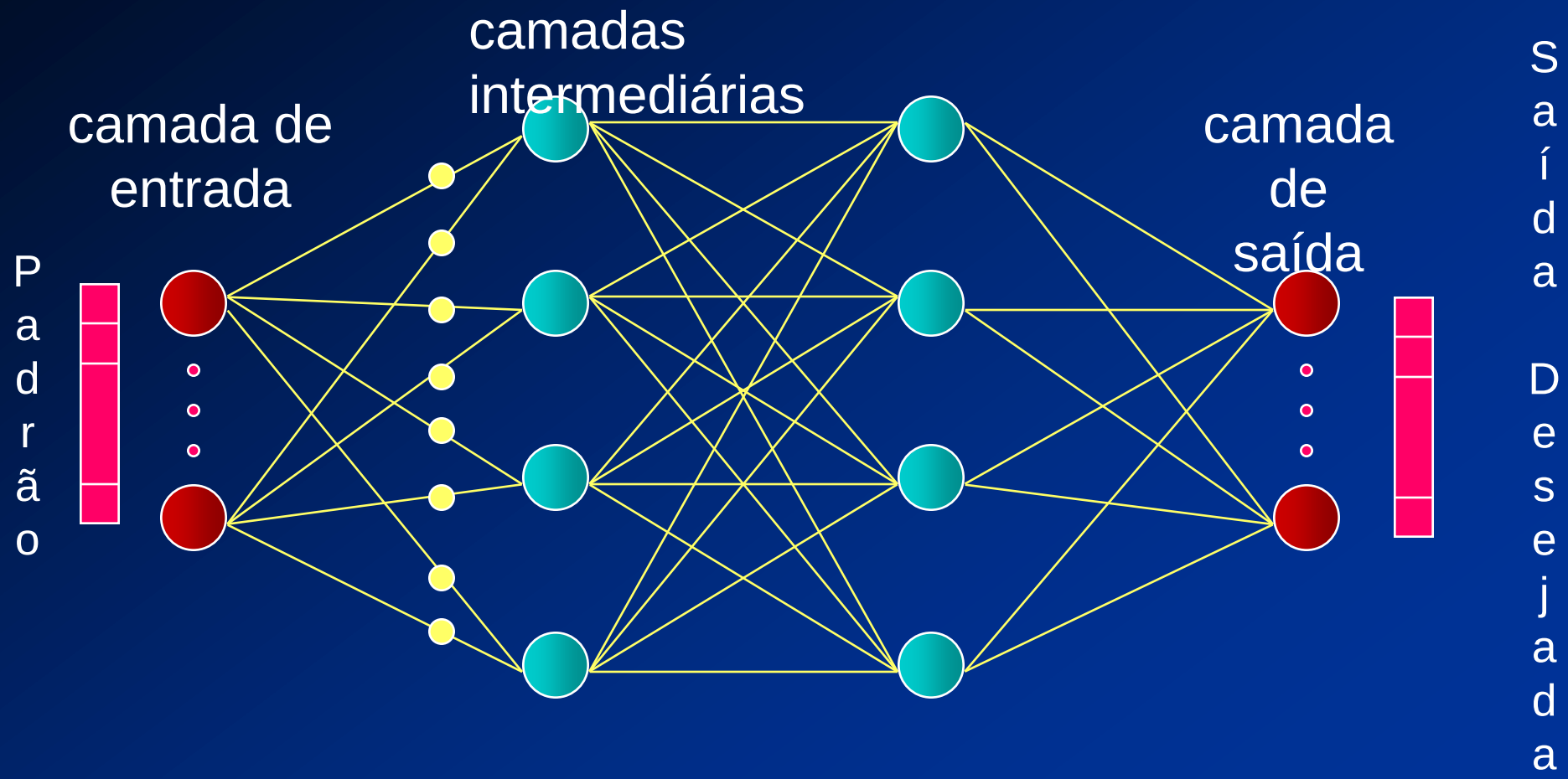
RNA - Aprendizado



RNA - Aprendizado



RNA - Aprendizado



Fase forward

- Entrada é apresentada à primeira camada da rede
  - Após os neurônios da camada *i* calcularem seus sinais de saída, os neurônios da camada *i + 1* calculam seus sinais de saída
- Saídas produzidas pelos neurônios da última camada são comparadas às saídas desejadas
- Erro para cada neurônio da camada de saída é calculado

Fase backward

- A partir da última camada
  - O nó ajusta seu peso de modo a reduzir o seu erro
  - Nós das camadas anteriores tem seu erro definidos por:
    - Erros dos nós da camada seguinte conectados a ele ponderados pelos pesos das conexões entre eles

Backpropagation

- Treina redes MLP produzindo representações internas necessárias para nós intermediários
- Supor que cada combinação de pesos e thresholds corresponda a um ponto em uma superfície de solução
  - Solução = pontos mais baixos da superfície
  - Procura minimizar erro ajustando pesos e thresholds para que eles correspondam aos pontos mais baixos da superfície
    - método do gradiente descendente

Backpropagation

- Gradiente de uma função está na direção e sentido onde a função tem taxa de variação máxima
  - Garantido de achar uma solução para superfícies simples
- Backpropagation fornece aproximação da trajetória no espaço de peso computado pelo método do gradiente descendente

Backpropagation

- Processamento
  - Forward (teste)
  - Backward (treinamento)
- Estados de ativação
  - 1 (+1) = ativo
  - 0 (-1) = inativo

Backpropagation

- Função de ativação
  - Não linear
  - Diferenciável , contínua e, geralmente, não decrescente
  - Sigmoidal
  - $a_i(t+1) = 1/(1 + e^{-u_i(t)})$ (sigmoidal logística)
  - $a_i(t+1) = \dfrac{(1 - e^{-u_i(t)})}{(1 + e^{-u_i(t)})}$ (tang. hiperbólica)

Backpropagation

- Funcionamento do algoritmo:
  - Ponto de partida para obter a expressão de ajuste de pesos:

$$E = \frac{1}{2}\sum_j (d_j - y_j)^2$$

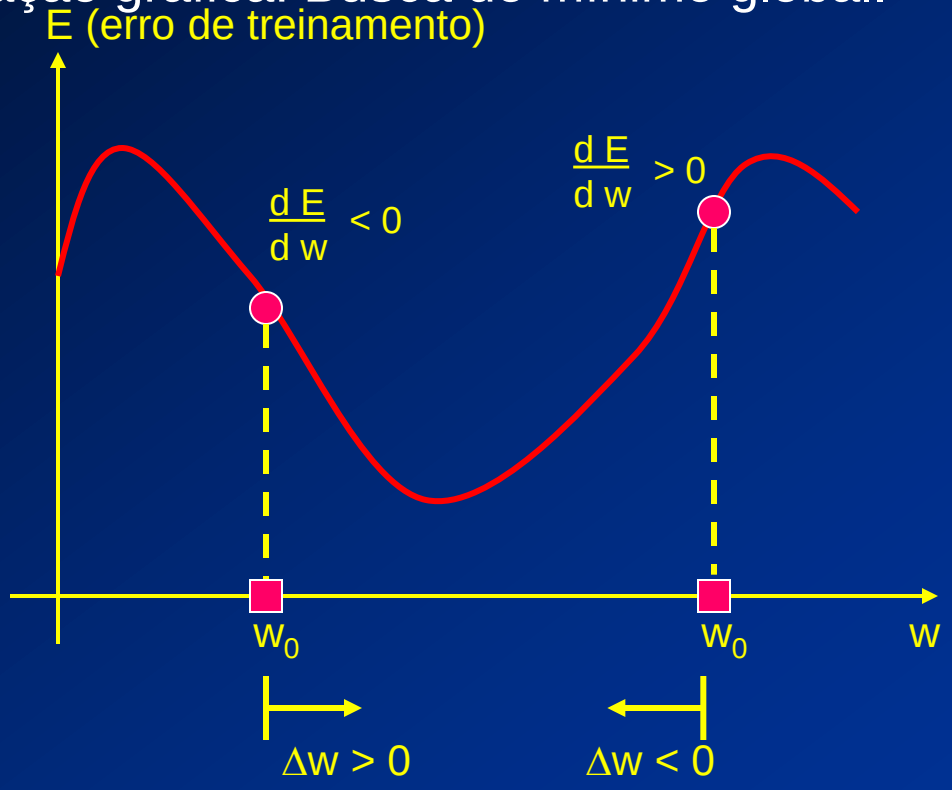
Erro para um padrão, considerando todos os nodos de saída.

$$\Delta w_{ji} \propto -\frac{\partial E}{\partial w_{ji}}$$

Para caminhar em direção ao mínimo, o peso é ajustado com sinal contrário ao da derivada.

Backpropagation

- Interpretação gráfica: Busca do mínimo global.



Backpropagation

- A partir desta idéia, fazendo manipulações matemáticas, obtemos a expressão de ajuste de

$$w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji}$$

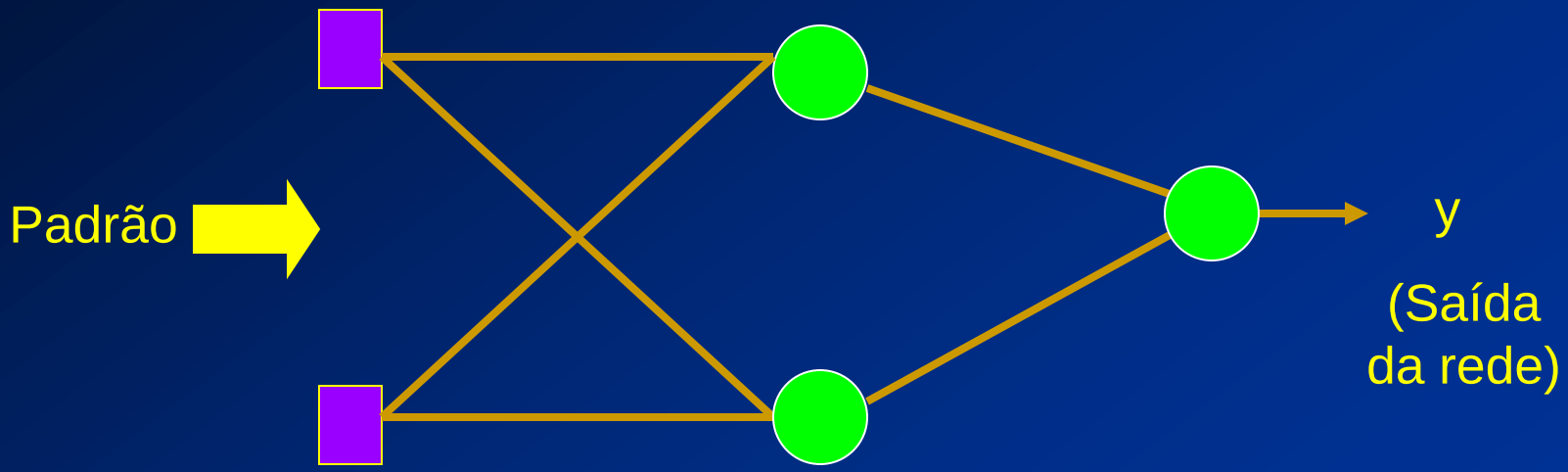
$$onde \quad \Delta w_{ji} = \eta\, \delta_j(t)\, x_i(t)$$

$$onde \quad \delta_j(t) = \begin{cases} (d_j - y_j)\; f'(net_j), & se\ for\ nodo\ de\ saída \\ (\sum_l \delta_l\; w_{lj})\; f'(net_j), & caso\ contrário \end{cases}$$

- Em cada iteração, o algoritmo realiza duas fases:
  - Forward (a rede gera suas saídas a partir das entradas),
  - Backward (a rede ajusta seus pesos a partir das saídas).

Backpropagation

Fase Forward: Apresenta-se o padrão à rede, que gera uma saída.

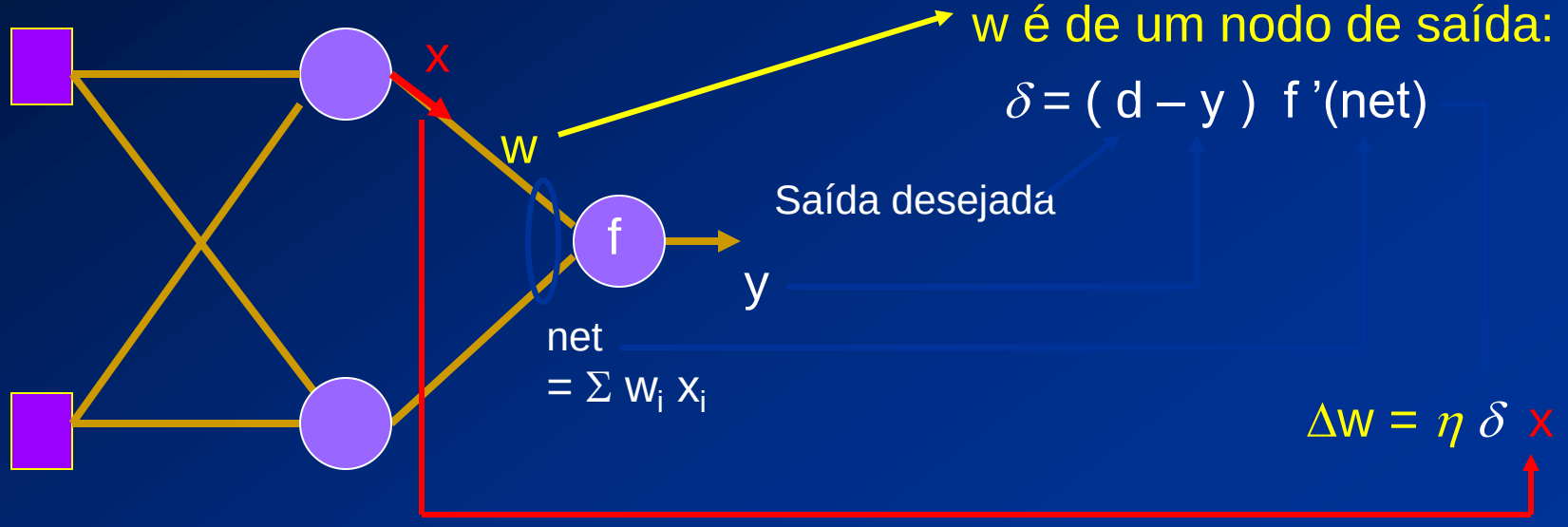


Backpropagation

Fase Backward: Ajusta os pesos da rede a partir da camada de saída.

$$\Delta w_{ji} = \eta\, \delta_j(t)\, x_i(t)$$

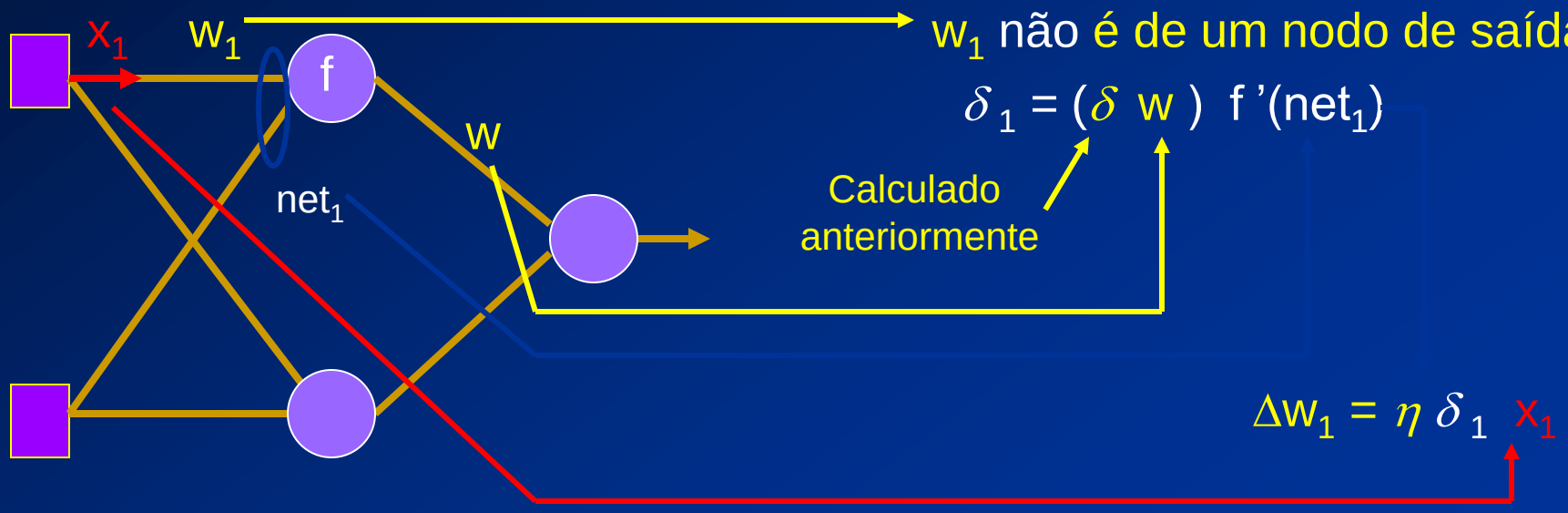
$$onde \quad \delta_j(t) = \begin{cases} (d_j - y_j)\, f'(net_j), & se\ for\ nodo\ de\ saída \\ (\sum_l \delta_l\, w_{lj})\, f'(net_j), & caso\ contrário \end{cases}$$



Backpropagation

$$\Delta w_{ji} = \eta\, \delta_j(t)\, x_i(t)$$

$$\text{onde} \quad \delta_j(t) = \begin{cases} (d_j - y_j)\, f'(net_j), & \text{se for nodo de saída} \\ (\sum_l \delta_l\, w_{lj})\, f'(net_j), & \text{caso contrário} \end{cases}$$



Treinamento

1) Iniciar todas as conexões com valores aleatórios

2) Repita

erro = 0

Para cada par de treinamento (X, d)

Para cada camada k := 1 a N

Para cada neurônio J := 1 a M_k

Calcular a saída y_{jk}

Se erro > ε

Então Para cada camada k := N a 1

Para cada neurônio J := 1 a M_k

Atualizar pesos

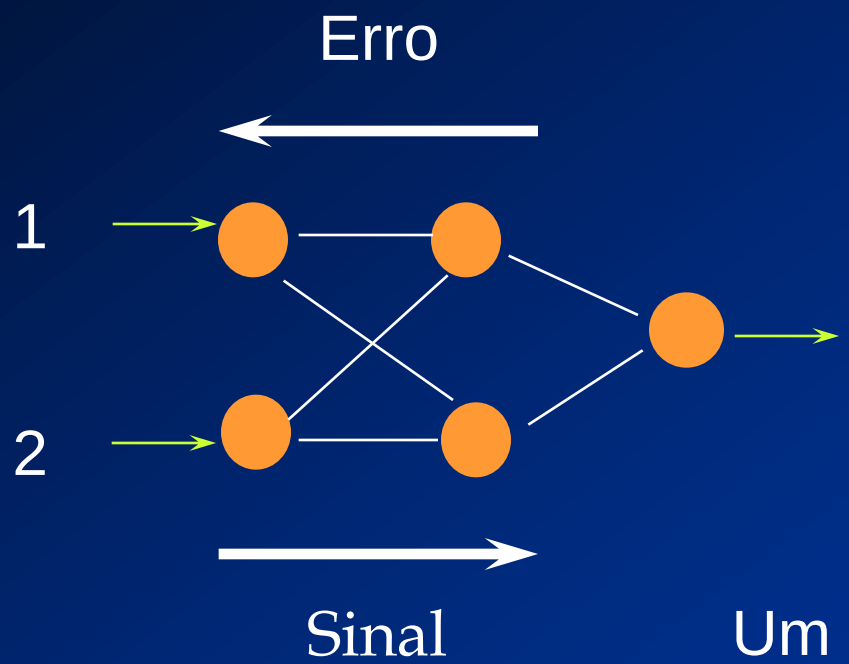
Até erro < ε

Teste

1) Apresentar padrão X a ser reconhecido
2) Para cada camada k := 1 a N
   Para cada neurônio J := 1 a M_k
   Calcular a saída y_{jk}
   Comparar saída y_{Nj} com d_{cj} para cada classe c
   Classificar padrão como pertencente a classe cuja saída desejada é mais próxima da saída produzida

Exemplo de rede

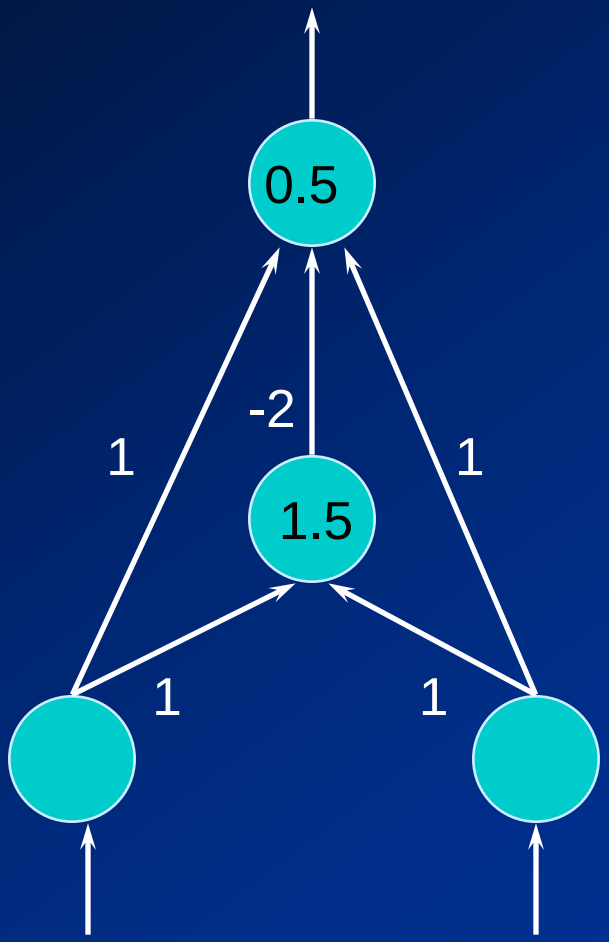
- Rede que aprende a função:



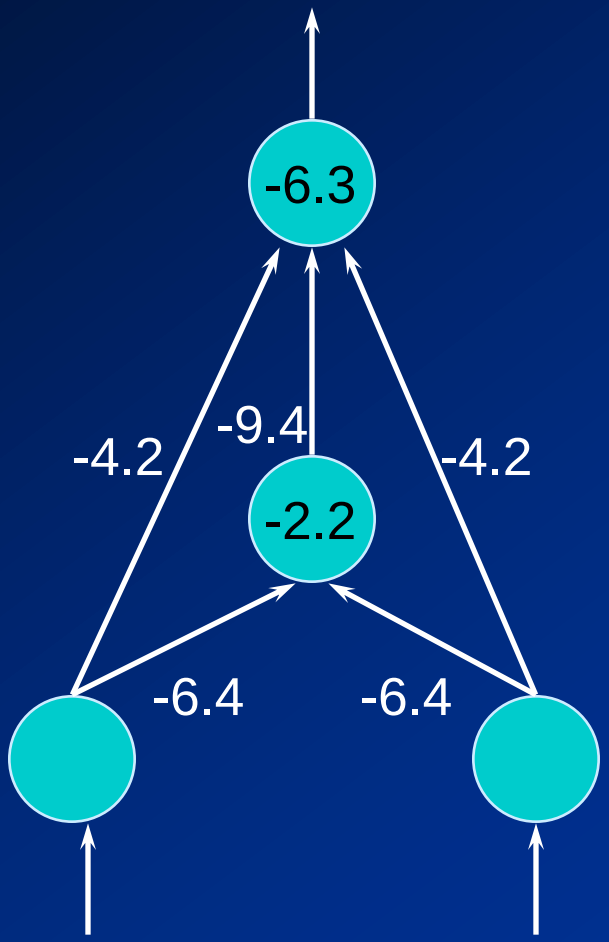
Ent. 1	Ent. 2	Saída
V	V	F
V	F	V
F	V	V
F	F	F

Um sinal verdadeiro é codificado para 1 e um falso para 0

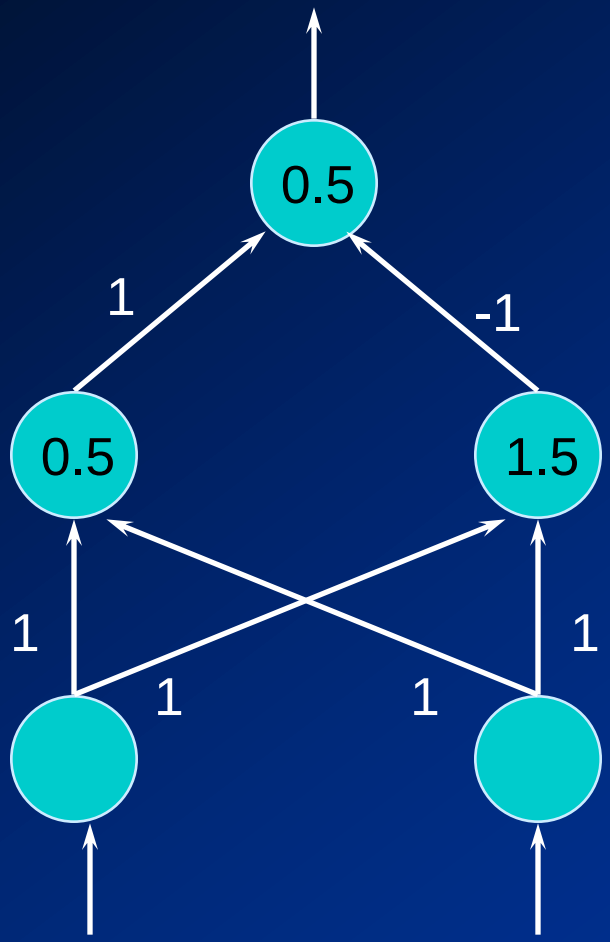
Possível solução



Solução após treinamento

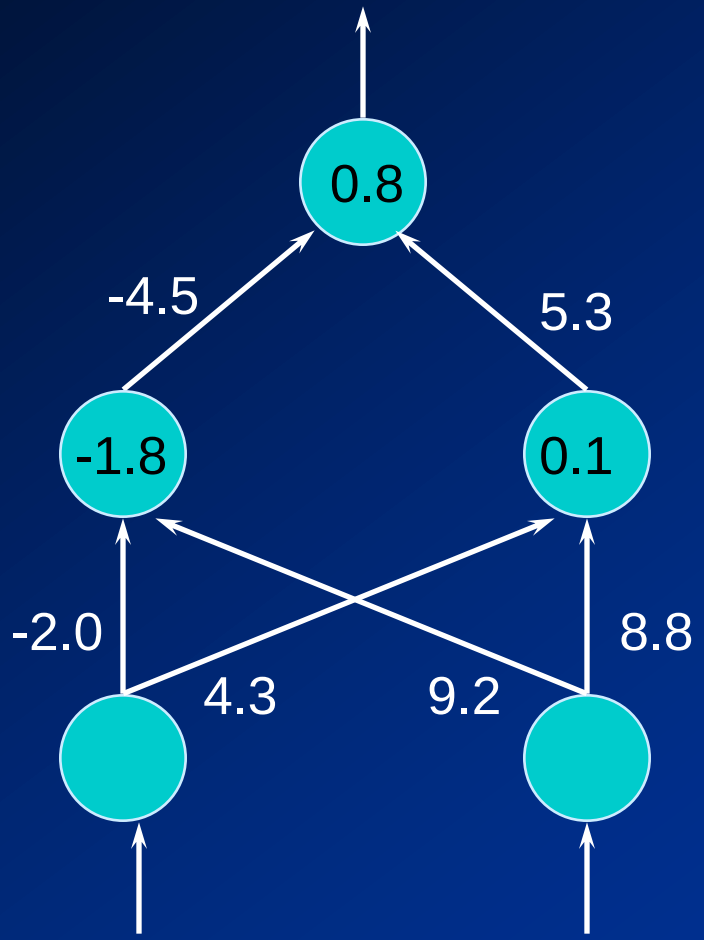


Utilizando uma rede diferente



Rede sem conexões entrada-saída

Problemas



Rede estável que não funciona

Mínimo local

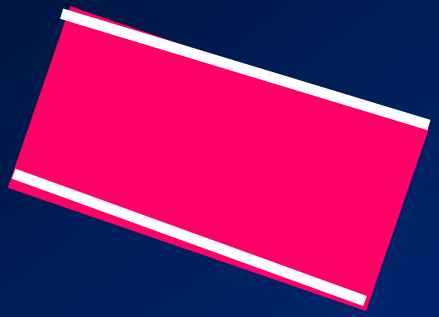
Ocorre em 1% das vezes para problemas do tipo ou-exclusivo

MLPs como classificadores

- Função implementada por cada neurônio é formada pela combinação das funções implementadas por neurônios da camada anterior
  - Camada 1: linhas retas no espaço de decisão
  - Camada 2: regiões convexas
    - Número de lados = número de unidades na camada 1
  - Camada 3: Combinações de figuras convexas, produzindo formatos abstratos

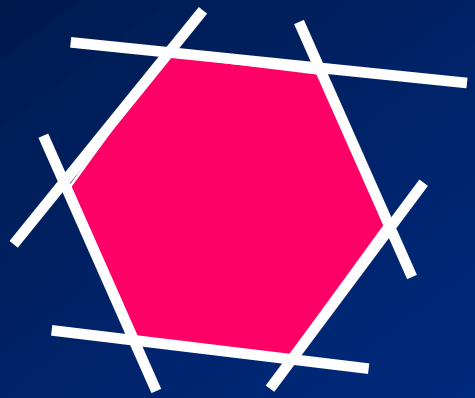
Regiões convexas

Aberta

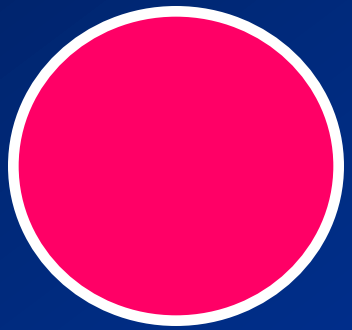


Aberta

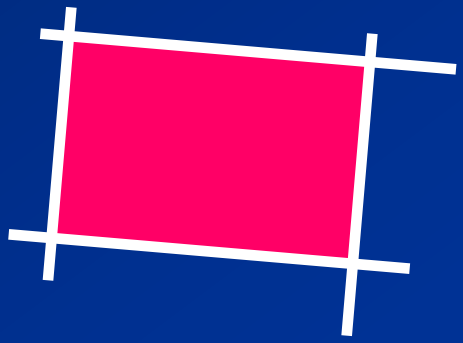
Fechada



Fechada

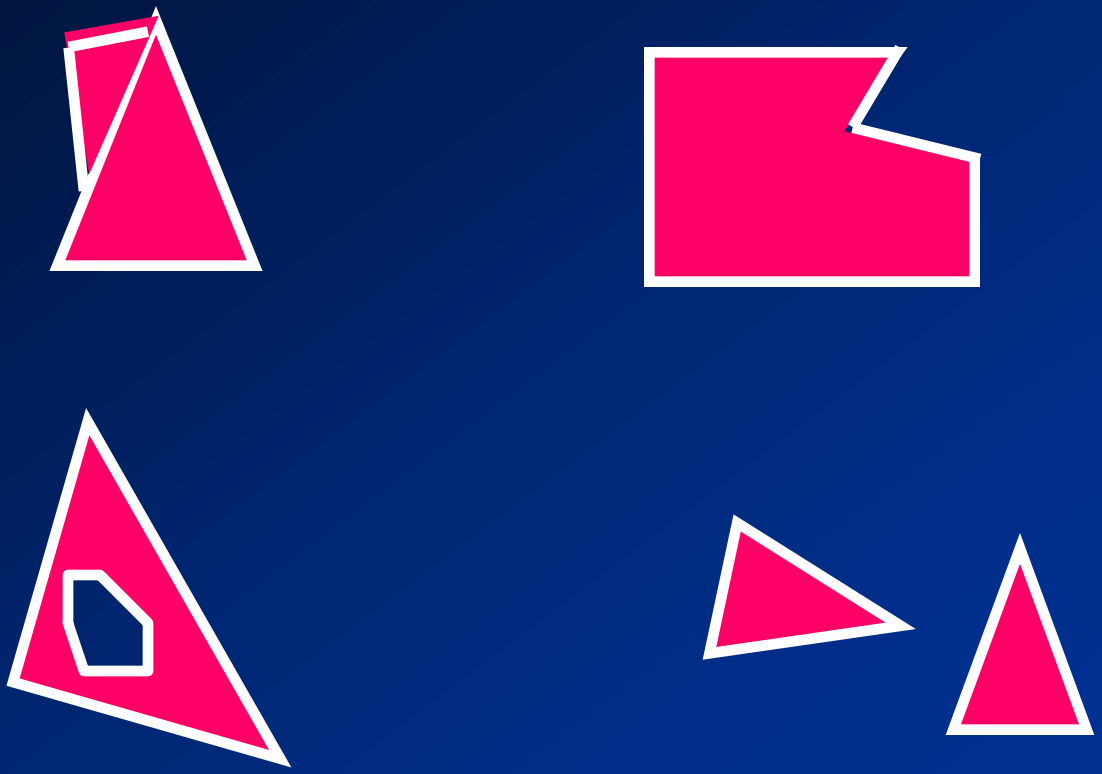


Fechada



Fechada

Combinações de regiões convexas



Unidades intermediárias

- Número de camadas intermediárias necessárias
  - 1 camada: suficiente para aproximar qualquer função contínua ou Booleana
  - 2 camadas: suficiente para aproximar qualquer função
  - 3 ou mais camadas: pode facilitar o treinamento da rede
    - Cada vez que o erro é propagado para a camada anterior, ele se torna menos útil

Unidades intermediárias

- Número de neurônios nas camadas intermediárias
  - Em geral não é conhecido
  - Utilizar função do número de entradas e saídas
    - Não funciona
  - Número de pesos vezes dez é menor que o número de exemplos
    - Apenas reduz overfitting
    - Se o número de exemplos for muito maior que o número de pesos, overfitting é improvável, mas pode ocorrer underfitting

Unidades intermediárias

- Número de neurônios nas camadas intermediárias (cont.)
  - Depende de:
    - Número de exemplos de treinamento
    - Quantidade de ruído
    - Complexidade da função a ser aprendida
      - Distribuição estatística

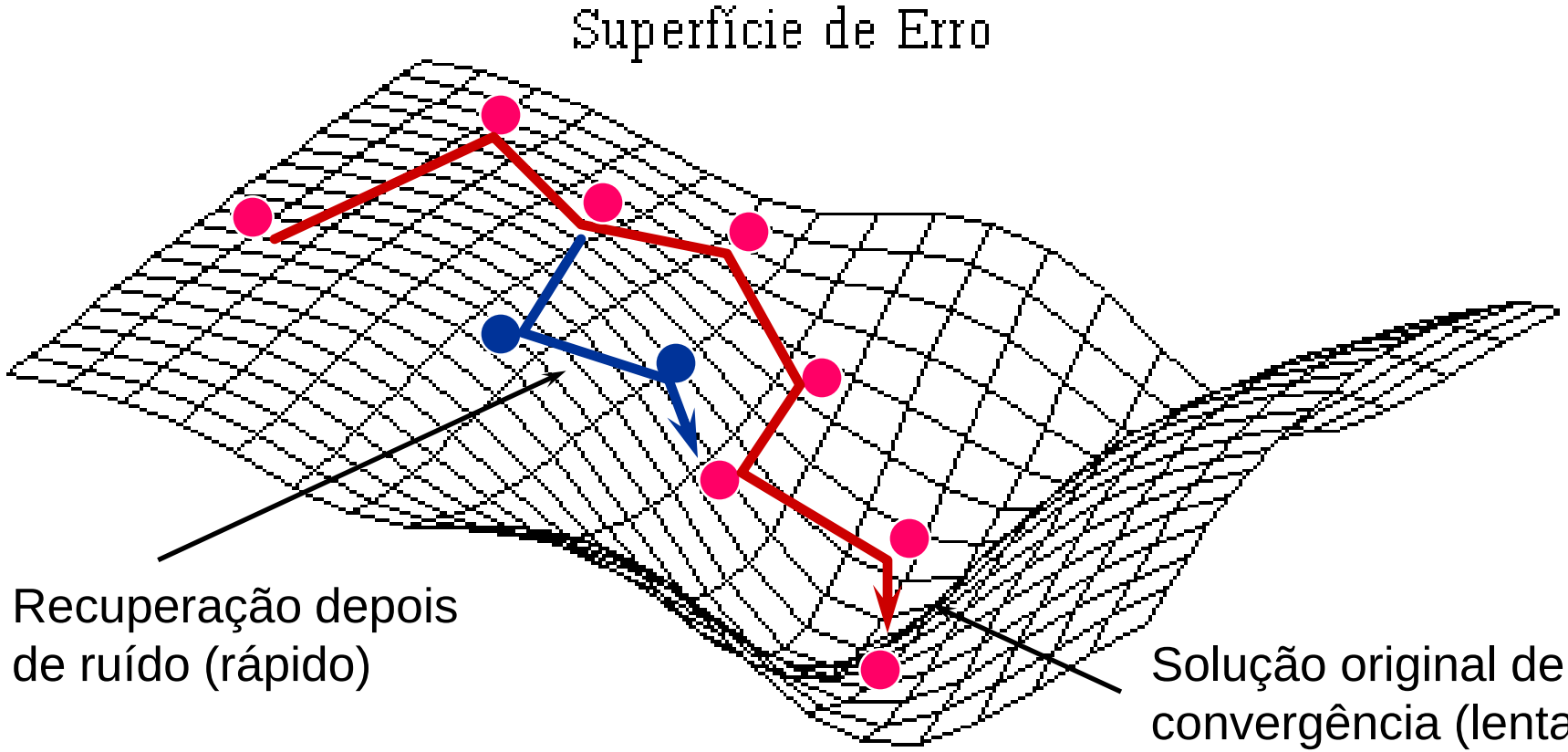
Unidades intermediárias

- Número de neurônios nas camadas intermediárias (cont.)
  - Existem problemas com uma entrada e uma saída que precisam de milhares de unidades e vice-versa
  - Pode crescer exponencialmente com o número de entradas
  - Solução neural eficiente: aquela onde o número de unidades cresce apenas polinomialmente com o número de entradas

Generalização

- Classificação correta de padrões não utilizados no treinamento ou com ruído
- Ocorre através da detecção de características relevantes do padrão de entrada
- Padrões desconhecidos são atribuídos a classes cujos padrões apresentam características semelhantes
- Tolerância a falhas

Generalização



Dificuldades de aprendizado

- Backpropagation é muito lento em superfícies complexas
  - Considerar efeitos de segunda ordem para gradiente descendente
- Mínimos locais: solução estável que não fornece saída correta
  - Taxa de aprendizado decrescente
  - Adicionar nós intermediários
  - Utilizar momentum
  - Adicionar ruído

Dificuldades de aprendizado

- Overfitting
  - Depois de um certo ponto do treinamento, a rede piora ao invés de melhorar
  - Memoriza padrões de treinamento, incluindo suas peculiaridades (piora generalização)
  - Alternativas
    - Encerrar treinamento cedo
    - Reduzir pesos

Atualização dos pesos

- Ciclo
  - Apresentação de todos os exemplos de treinamento durante o aprendizado
  - Exemplos devem ser apresentados em ordem aleatória
- Abordagens para atualização dos pesos
  - Por padrão (online)
  - Por ciclo (batch)

Atualização dos pesos

- Por padrão
  - Pesos atualizados após apresentação de cada padrão
  - Estável se taxa de aprendizado e momentum forem pequenos (reduzir progressivamente as taxas)
  - Altas taxas $\Rightarrow$ rede instável
  - Mais rápida, principalmente se o conjunto de treinamento for grande e redundante
  - Requer menos memória

Atualização dos pesos

- Por ciclo
  - Pesos atualizados depois que todos os padrões de treinamento forem apresentados
  - Geralmente mais estável
  - Pode ser lento se o conjunto de treinamento for grande e redundante
  - Estimativa mais precisa do vetor gradiente
- Método depende da aplicação

Atualização dos pesos

- Momentum
  - $\Delta w_{ij}(t + 1) = \eta x_i y_j (1 - y_j)\delta_j + \alpha(w_{ij}(t) - w_{ij}(t - 1))$
  - Aumenta velocidade de aprendizado evitando perigo de instabilidade
  - Pode acelerar treinamento em regiões muito planas da superfície de erro
  - Suprime oscilação de pesos em vales e ravinas

Dicas para melhorias

- Projeto de uma RNA utilizando backpropagation é mais uma arte que uma ciência
  - Envolve inúmeros fatores
  - Resultado da experiência do projetista
- Utilizar função sigmoidal assimétrica (tangente hiperbólica)
  - Aprendizado mais rápido (em geral)
  - Igual a função logística com bias e re-escalada

Dicas para melhorias

- Resposta desejada deve estar [-a + ε, a - ε]
  - a = valor máximo da função de ativação
- Inicialização dos pesos e thresholds deve ser uniformemente distribuído dentro de um intervalo pequeno
  - Reduz probabilidade dos neurônios saturarem
  - Intervalos muito pequenos podem tornar treinamento lento
  - Geralmente utiliza-se (-2.4/fan_in, + 2.4/fan_in)

Dicas para melhorias

- Taxa de aprendizado não deve, preferencialmente, ser a mesma para todos os neurônios
  - Geralmente, últimas camadas têm gradiente maior que camadas iniciais
    - Taxa de aprendizado deve ser menor para neurônios das últimas camadas
  - Neurônios com muitas entradas, devem ter taxas de aprendizado menores

Dicas para melhorias

- Utilizar, sempre que possível, modo padrão
  - Atualização *on-line*
  - Classificação de padrões envolvendo base da dados grande e redundante
  - Paralelização ineficiente
- Incluir conhecimento a priori
- Utilizar técnicas de prunning
  - Elimina conexões não essenciais
- Empregar cross-correlation

Aplicações

- Gerar sons a partir de textos
- Reconhecimento de padrões
- Venda de passagens aéreas
- Filtragem de ruído de eletrocardiograma
- Previsão de séries temporais

Conclusão

- Necessidade de redes multicamadas
  - Redes de uma camada não resolvem problemas não linearmente separáveis
  - Problema: treinamento de redes multicamadas
  - Backpropagation
    - Overfitting
    - Momento