IF-705 – Automação Inteligente
Perceptrons de Múltiplas Camadas

Aluizio Fausto Ribeiro Araújo
Universidade Federal de Pernambuco
Centro de Informática - CIn
Departamento de Sistemas da Computação
aluizioa@cin.ufpe.br

Conteúdo

- Histórico
- Introdução
- Perceptron de Camadas Múltiplas
- Algoritmo de Retropropagação: Descrição, Funções, Capacidades e Exemplos
- Arquivo de Treinamento para Generalização.
- Seleção de Modelo
- Protocolo de Treinamento

Histórico

- O perceptron de Rosenblatt: Neurônio com pesos e bias ajustáveis.
- Teorema da convergência do perceptrom: Se padrões empregados para treinar o perceptron são extraídos de duas classes linearmente separáveis, então o algoritmo converge e posiciona uma superfície de decisão na forma de hiperplano para separar duas classes.
- Limitações do perceptron:
  - Resolve apenas casos linearmente separáveis.
  - A não-linearidade na ativação do perceptron (função limiar) não é diferenciável, logo não pode ser aplicada a camadas múltiplas.
- Um simples neurônio gera uma classe de soluções que não são gerais para resolver problemas mais complexos, justificando o aparecimento do Perceptron de Camada Múltipla (MLP).

Introdução

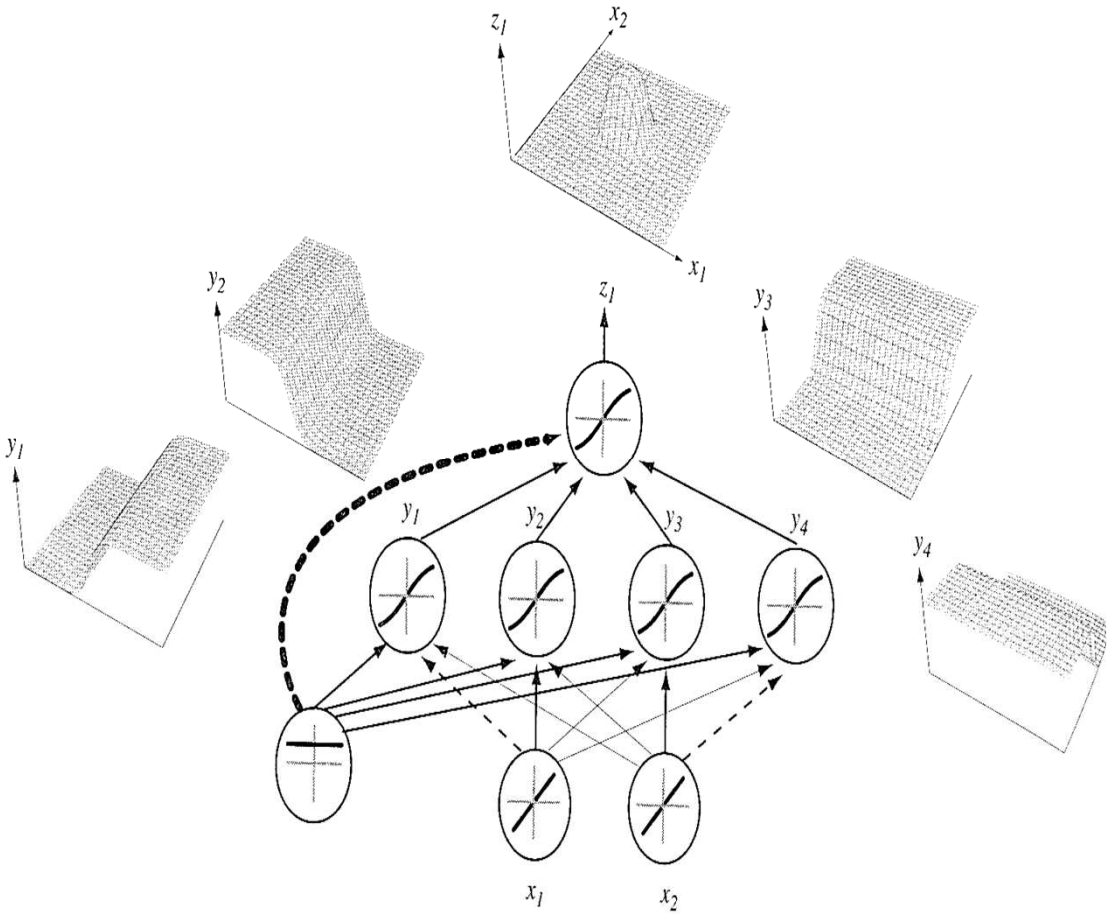
- A rede neural supervisonada chamada Perceptron multicamadas (*multilayer percetron*) utiliza métodos derivados do gradiente no ajustes de seus pesos por retropropagação.
  - Esta rede consiste de uma camada de entrada, uma ou mais camadas escondidas e uma camada de saída. Um sinal de entrada é propagado, de camada em camada, da entrada para a saída.
  - MLP é treinada com um algoritmo de retropropagação do erro.
- Estágios da aprendizagem por retropropagação do erro:
  - Passo para frente: Estímulo aplicado à entrada é propagado para frente até produzir resposta da rede.
  - Passo para trás: O sinal de erro da saída é propagado da saída para a entrada para ajuste dos pesos sinápticos.

Introdução

- A MLP apresenta três características importantes:
  - Cada unidade de processamento tem função de ativação logística (forma sigmoidal) que é não-linear, suave e diferenciável.
  - Existência de ao menos uma camada escondida que possibilita aprendizagem de tarefas complexas por extração progressiva de características relevantes dos padrões de entrada.
  - O grau de conectividade é alto.
- As limitações de um Percepton simples devem desaparecer quando se utiliza camadas intermediárias, ou escondidas, entre as camadas de entrada e de saída.
- O emprego de Redes MLP aumentou com a introdução do método retropropagação para aprendizagem.

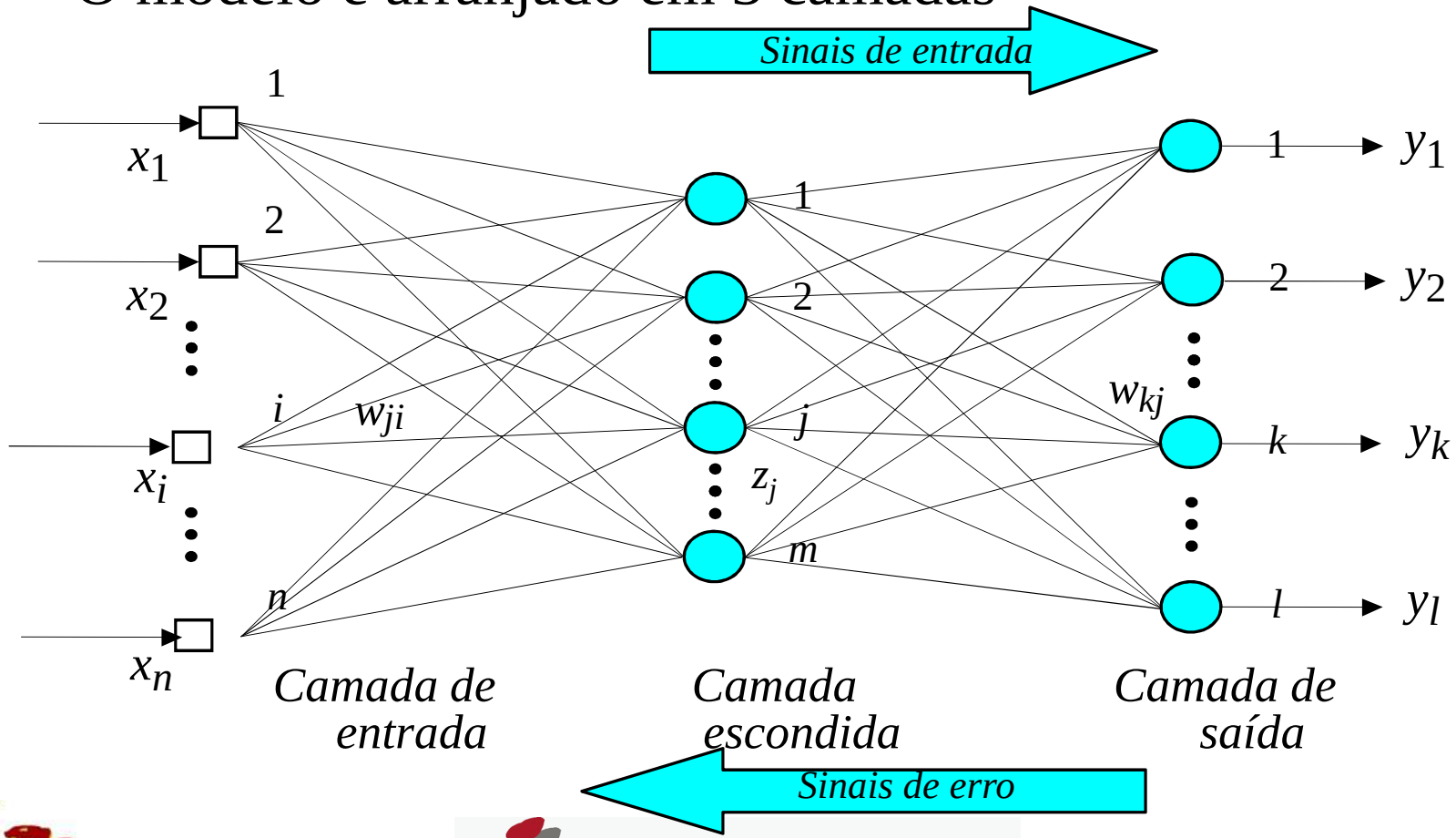
Introdução

- Sumário da MLP-RP
  - Camada de entrada
  - Camada escondida
  - Camada de saída
  - Unidade de bias
  - $net_j = \mathbf{w}^t_j \mathbf{x}$
  - $y_j = f(net_j)$
  - $f(net_k) = \text{Sigm}(net_k)$
  - $z_k = f(net_k)$



Perceptron de Camadas Múltiplas

- O modelo é arranjado em 3 camadas



Algoritmo MLP-BP

- Sinal de erro para a unidade *k* na iteração *n*: $e_k(n) = t_k(n) - y_k(n)$
- Erro total:
  - *C* é o conjunto de nodos de saída.

$$E(n) = \frac{1}{2}\sum_{k \in C} e_k^2(n)$$

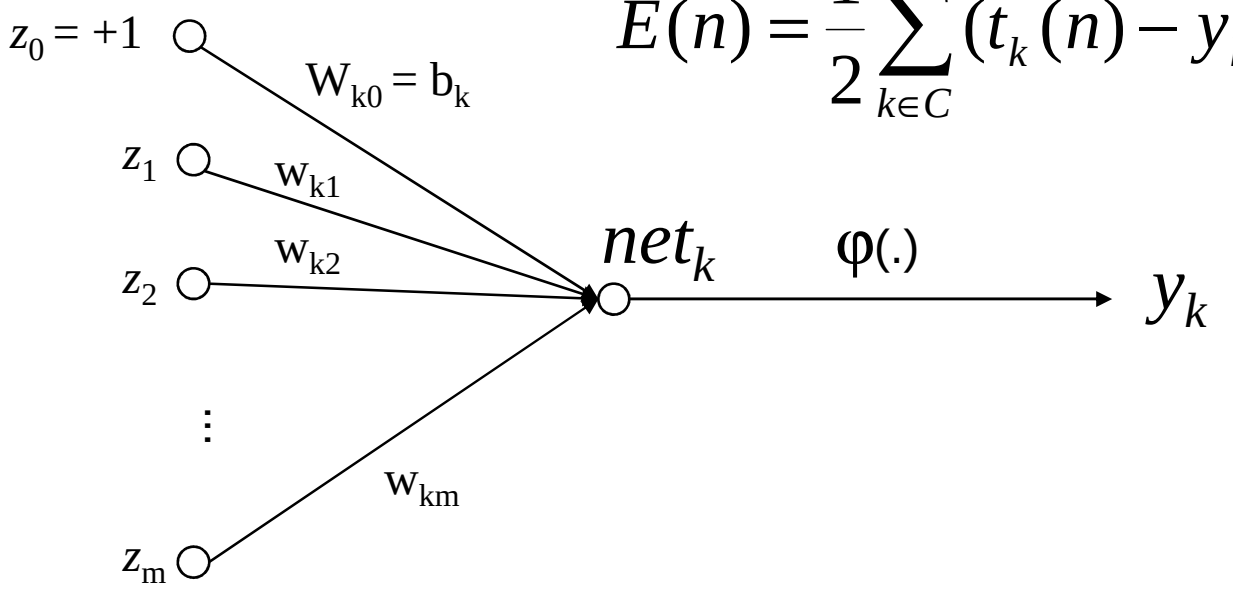
- Erro quadrado médio:
  - Média sobre todo conjunto de treinamento.
  - *Npad* é o número de padrões.

$$E_{av} = \frac{1}{N_{pad}}\sum_{n=1}^{Npad} E(n)$$

- Processo de aprendizagem: Ajusta parâmetros (pesos sinápticos) para minimizar *Eav*.
- Pesos são atualizados padrão a padrão até completar uma época (apresentação completa do conjunto de treinamento).

Algoritmo MLP-BP

- Notação:



$$E(n) = \frac{1}{2}\sum_{k \in C}(t_k(n) - y_k(n))^2$$

$$net_k(n) = \sum_{i=0}^{m} w_{kj}(n)\, z_j(n) \qquad y_k(n) = \varphi_k(net_k(n))$$

Algoritmo MLP-BP

- Atualização dos pesos

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \Delta\mathbf{W}(n)$$

- Gradiente Descendente

$$\Delta w_{pq}(n) = -\eta \frac{\partial E(n)}{\partial w_{pq}(n)}$$

- Por simplicidade de notação, o índice *n* será retirado.

$$\Delta w_{pq} = -\eta \frac{\partial E}{\partial w_{pq}}$$

MLP-BP: Ajuste de pesos *j*->*k* (nodo *h*-*o*)

- Gradiente determina a direção de busca no espaço de pesos.

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial net_k}\frac{\partial net_k}{\partial w_{kj}} = -\delta_k \frac{\partial net_k}{\partial w_{kj}} \therefore \frac{\partial net_k}{\partial w_{kj}} = \frac{\partial}{\partial w_{kj}}\sum_{q=1}^{m} w_{kq} z_q = \sum_{q=1}^{m}\frac{\partial w_{kq}}{\partial w_{kj}} z_q \therefore$$

$$\frac{\partial net_k}{\partial w_{kj}} = z_j, \quad \text{pois} \quad \frac{\partial w_{kq}}{\partial w_{kj}} = 0, \quad \frac{\partial w_{kj}}{\partial w_{kj}} = 1$$

- Sensibilidade descreve como o erro total se modifica com a atividade *net* de cada unidade.

$$\delta_k = -\frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial net_k} = (t_k - y_k)\varphi_k'(net_k)$$

  - Ajuste para nodo *h*-*o*: $\Delta w_{kj} = \eta\delta_k z_j = \eta(t_k - y_k)\varphi_k'(net) z_j$

MLP-BP: Ajuste de pesos *i*->*j* (nodo *i*-*h*)

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial z_j}\frac{\partial z_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}}, \qquad \text{onde} \qquad \frac{\partial E}{\partial z_j} = \frac{\partial}{\partial z_j}\left[\frac{1}{2}\sum_{k=1}^{l}(t_k - y_k)^2\right] \therefore$$

$$\frac{\partial E}{\partial z_j} = -\sum_{k=1}^{l}(t_k - y_k)\frac{\partial y_k}{\partial z_j} = -\sum_{k=1}^{l}(t_k - y_k)\frac{\partial y_k}{\partial net_k}\frac{\partial net_k}{\partial z_j} = -\sum_{k=1}^{l}(t_k - y_k)\varphi_k'(net_k)w_{kj}$$

$$\frac{\partial z_j}{\partial net_j} = \varphi_j'(net_j); \qquad \frac{\partial net_j}{\partial w_{ji}} = x_i;$$

$$\frac{\partial E}{\partial w_{ji}} = \left(-\sum_{k=1}^{l}(t_k - y_k)\varphi_k'(net_k)w_{kj}\right)\left(\varphi_j'(net_j)\right)(x_i) = \left(-\sum_{k=1}^{l}\delta_k w_{kj}\right)\left(\varphi_j'(net_j)\right)(x_i)$$

$$\Delta w_{ji} = \eta\delta_j x_i = \eta\left[\sum_{k=1}^{l} w_{kj}\delta_k\right]\varphi_j'(net_j)x_i$$

MLP-BP: Resumo da Aprendizagem

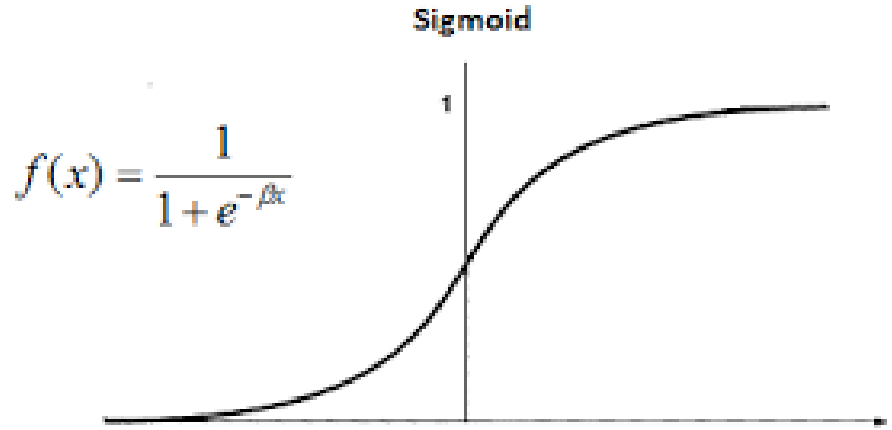
$$\begin{pmatrix}\text{correção}\\ \text{no peso}\\ \Delta w_{ji}(n)\end{pmatrix} = \begin{pmatrix}\text{taxa de}\\ \text{aprendizagem}\\ \eta\end{pmatrix} \cdot \begin{pmatrix}\text{sensibilidade}\\ \delta_j(n)\end{pmatrix} \cdot \begin{pmatrix}\text{sinal de entrada}\\ \text{para nodo } j\\ x_i(n)\end{pmatrix}$$

$$net_j(n) = \sum_{i=0}^{m} w_{ji}(n)\, x_i(n) \qquad y_j(n) = \varphi_j(net_j(n))$$

- Passagem entrada-saída
- Passagem saída-entrada
  - Compute recursivamente gradiente local δ
    - Da camada de saída para a de entrada
  - Modifique os pesos pela regra delta

MLP-BP: Funções de Ativação

- Função Sigmoide

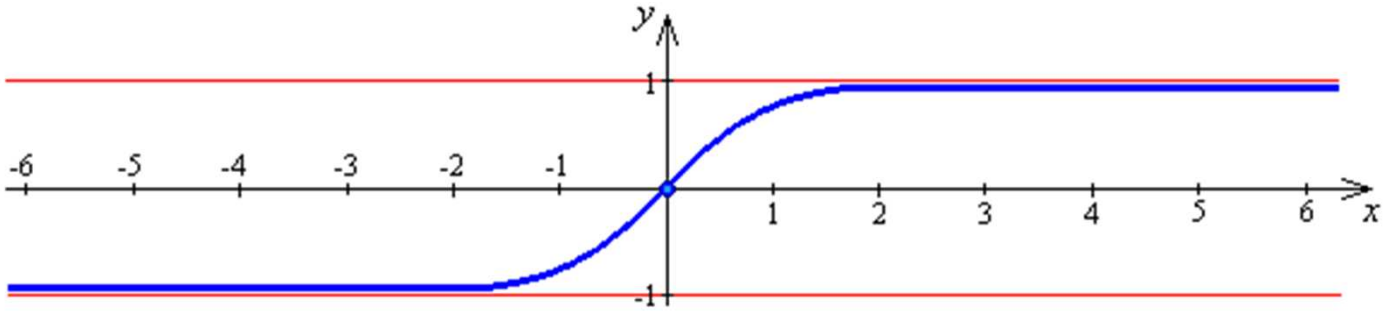


$$y_j = \varphi(net_j) = \frac{1}{1+\exp(-a \cdot net_j)} \quad a > 0 \text{ and } -\infty < net_j < \infty$$

$$\varphi'(net_j) = \frac{1}{1+\exp(-a \cdot net_j)} \frac{a \cdot [1+\exp(-a \cdot net_j) - 1]}{1+\exp(-a \cdot net_j)} = a \cdot y_j [1 - y_j]$$

MLP-BP: Funções de Ativação

- Função Tangente hiperbólica



$$y = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Domínio: $\mathbb{R}$

Imagem: $(-1, 1)$

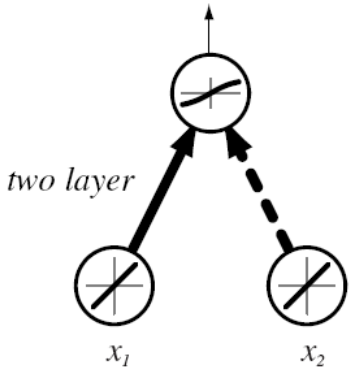
$$\lim_{x \to -\infty} \tanh(x) = -1 \quad \text{e} \quad \lim_{x \to +\infty} \tanh(x) = 1$$

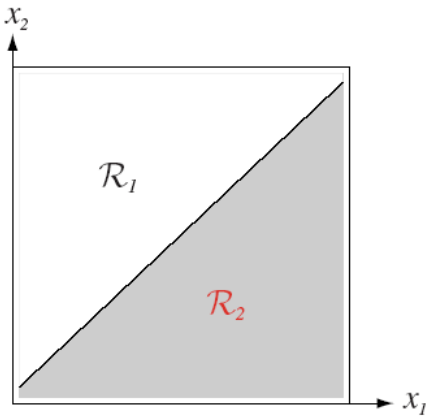
$$\varphi(net_i) = a \cdot \tanh(b \cdot net_j) \qquad (a,b) > 0$$

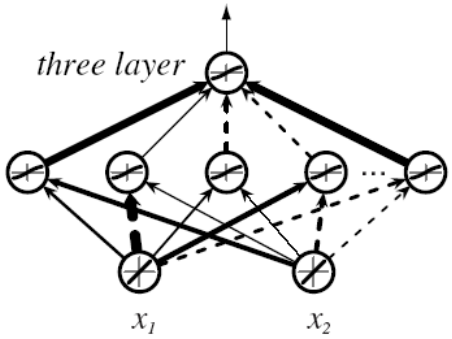
$$\varphi'(net_i) = a \cdot b \cdot \operatorname{sech}^2(b \cdot net_j) = a \cdot b(1 - \tanh^2(b \cdot net_j)) = \frac{b}{a}[a - y_j][a + y_j]$$

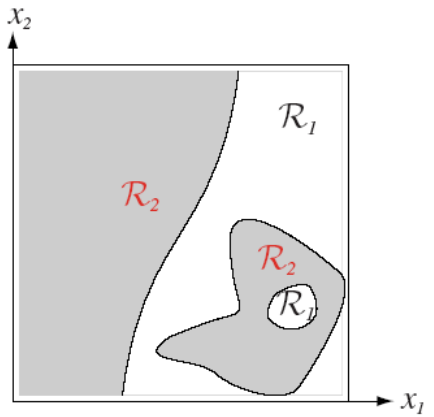
Poder Expressivo da MLP-PB

- Qualquer transformação pode ser implementada por 3 camadas?
  - Sim, qualquer função contínua da entrada para a saída pode ser implementada com número suficiente de nodos escondidos.
  - Prova-se pelo teorema de Kolmogorov.



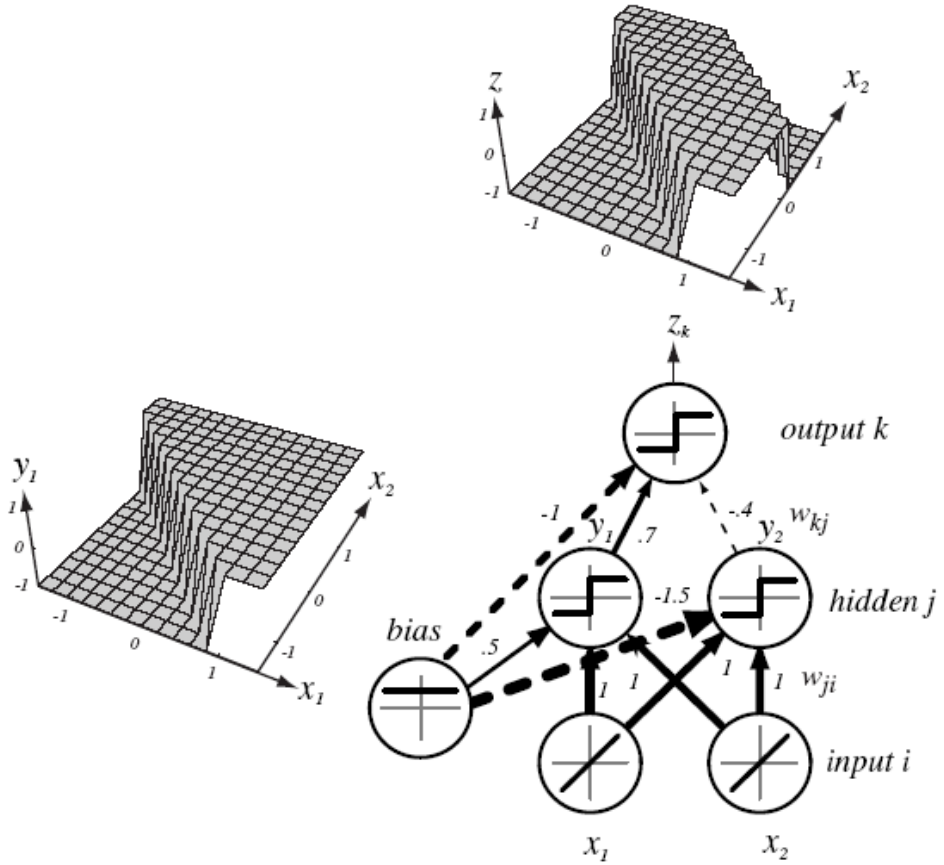




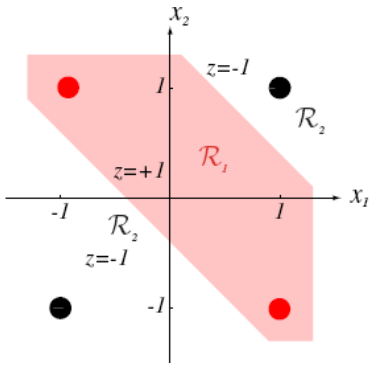


Poder Expressivo da MLP-PB

- Porta XOR: Operação da rede

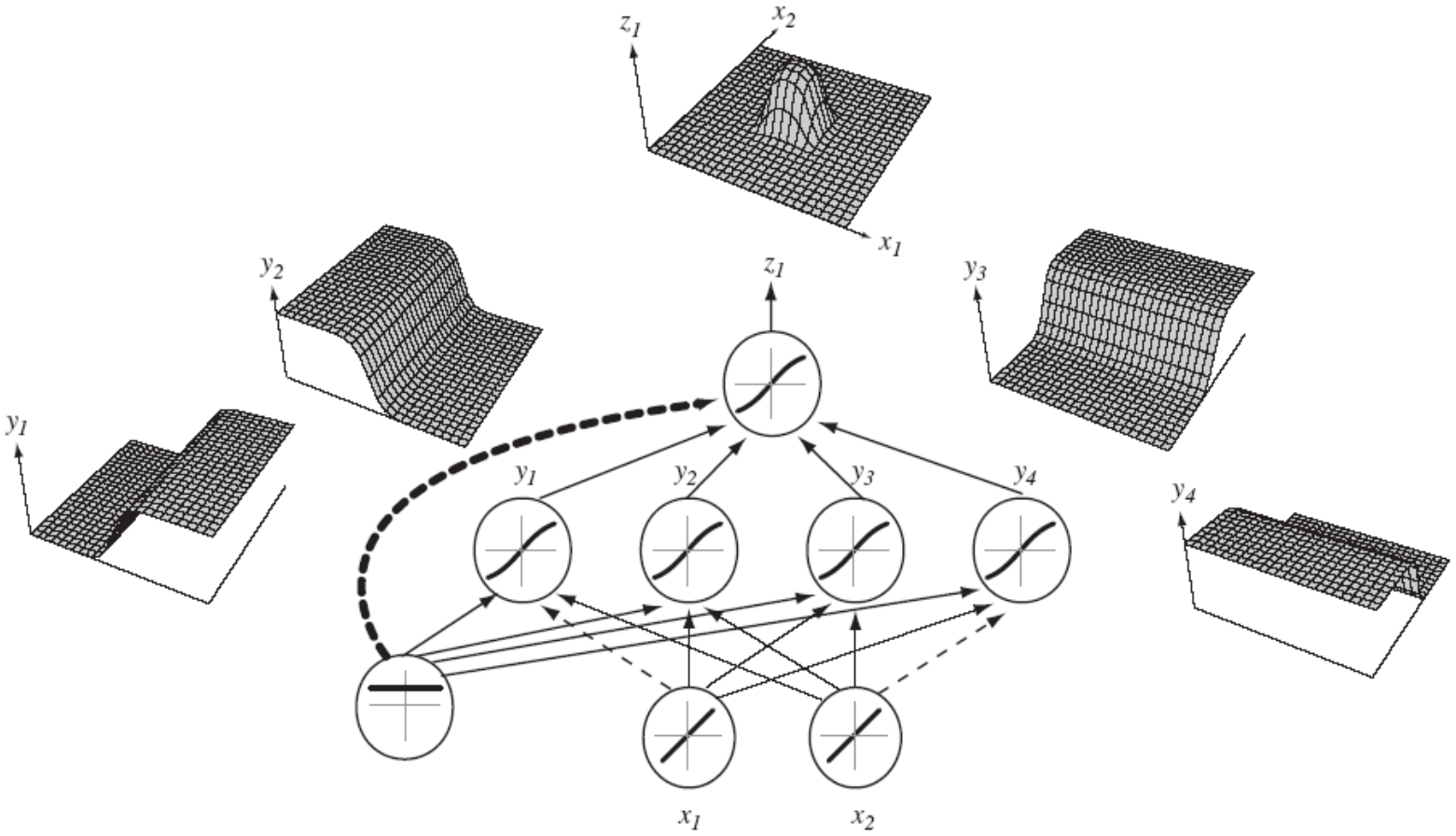


Divisão de classes



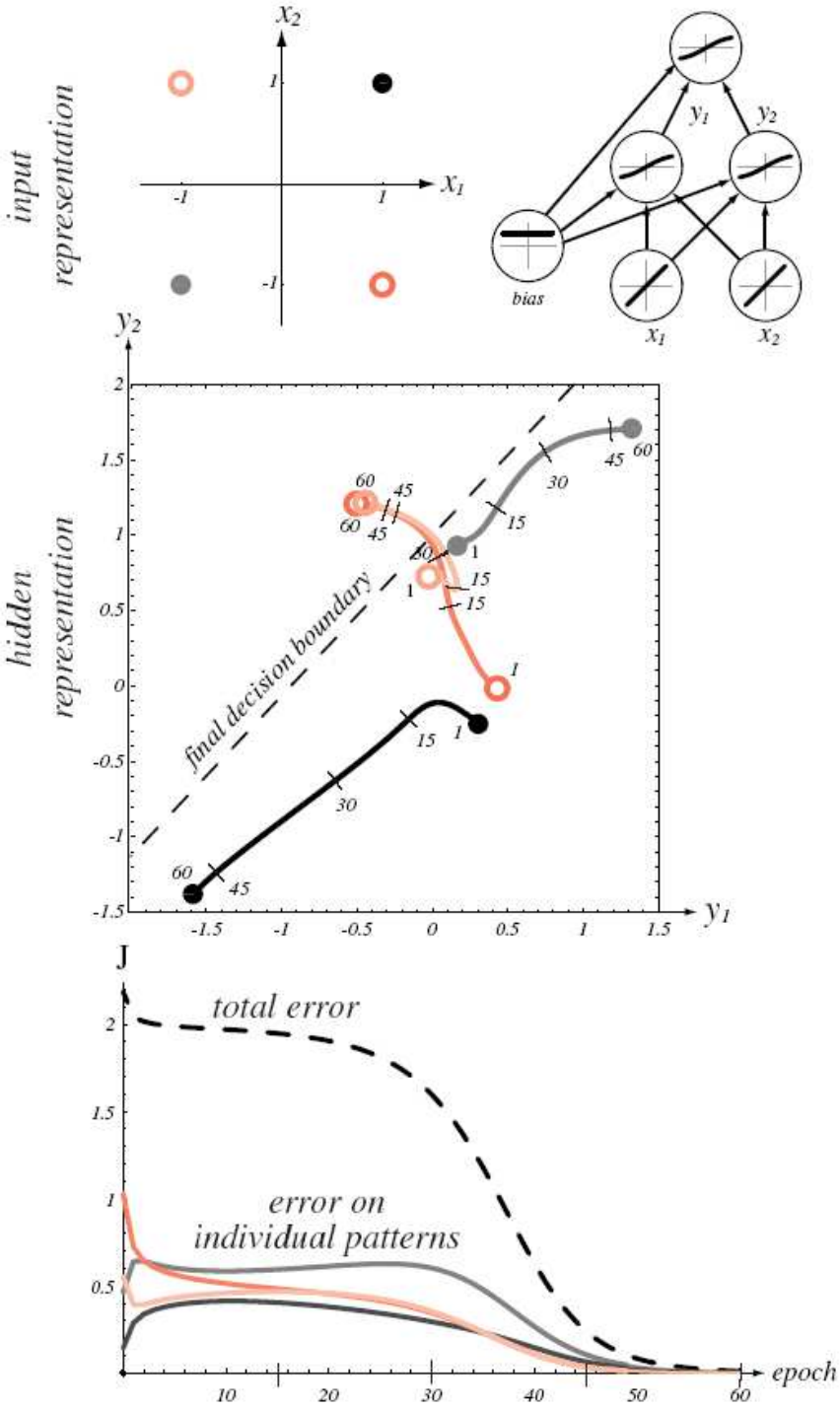
Poder Expressivo da MLP-PB

- Exemplo de operação



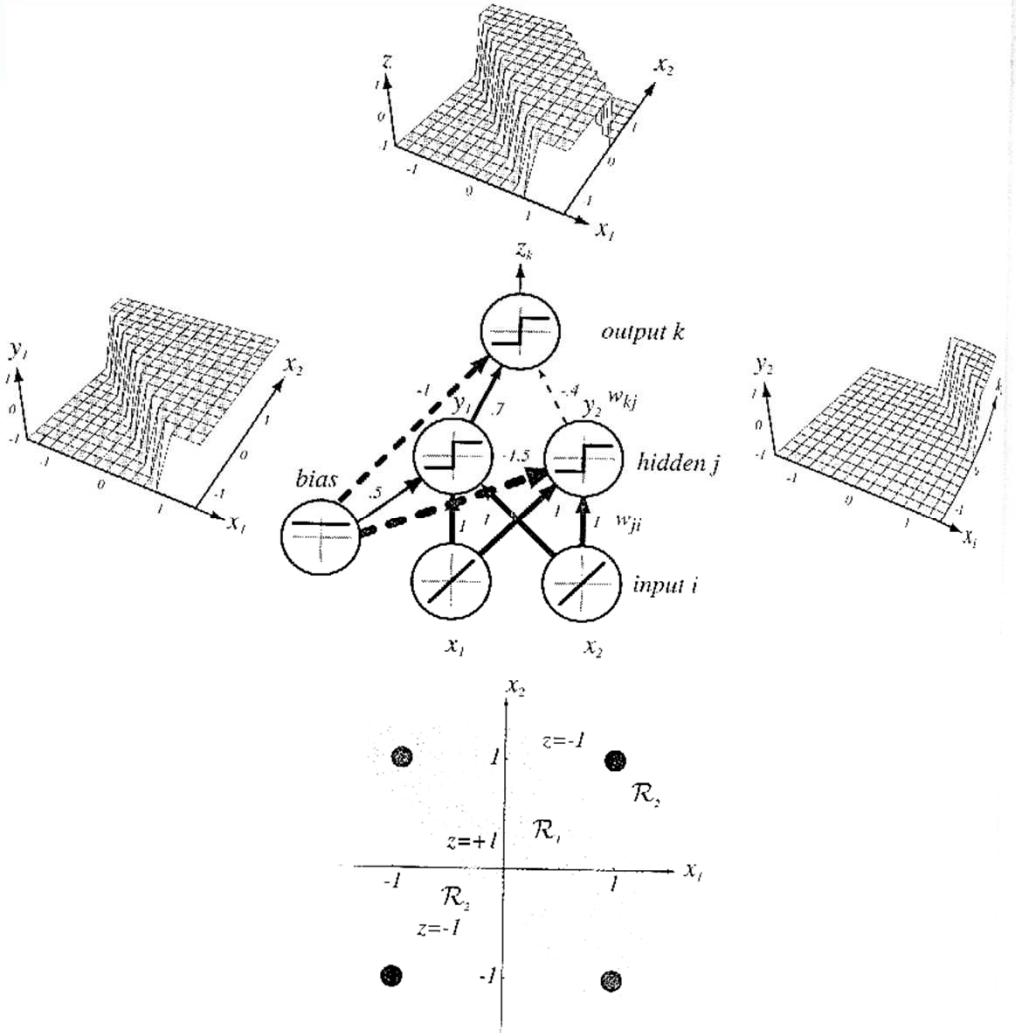
MLP-BP: Exemplo

- Detector de Características:
  - Unidades escondidas atuam como detectores de características.
  - O progresso da aprendizagem leva as unidades escondidas a descobrirem gradualmente característica salientes dos dados de treinamento.
  - Transformação não-linear do espaço de entrada para o de características.
  - Semelhança com o discriminante linear de Fisher.



MLP-BP: Exemplo

- **Porta XOR:**
  - (x_1 OR x_2) AND NOT (x_1 AND x_2)
  - $y_1 : x_1 + x_2 + 0.5 = 0$
  - > 0 -> $y_1 = 1$, de outro modo –1
  - $y_2 : x_1 + x_2 - 1.5 = 0$
  - > 0 -> $y_2 = 1$, de outro modo –1



Arquivo de Treinamento para Generalização

- Uma rede neural generaliza se seu mapeamento entrada-saída for completo ou aproximadamente correto para os dados de teste.
  - O processo de treinamento pode ser visto como a solução de um problema de ajuste de curva para interpolação não-linear dos dados de entrada.
  - Uma RN deixa de generalizar quando está sobretreinada, isto é, quando aprende muitos pares entrada-saída e passa a considerar características não gerais para o conjunto de padrões. Assim, ocorre memorização analogamente a uma tabela *look-up*.
  - É importante buscar mapeamentos não-lineares suaves (*smooth*) para demandar menos esforço computacional.

Seleção de Modelo

- Seleção de MLP-BP com o melhor número de pesos dados N amostras de treinamento. Deve-se encontrar *r*: para minimizar erro de classificação do modelo treinado pelo conjunto de estimação e testado pelo conjunto de validação.

$$r = \frac{\text{tamanho do conjunto de validação}}{\text{tamanho do conjunto de estimação} + \text{tamanho do conjunto de validação}}$$

  - Sugestão: $r = 0.2$
- Kearns(1996) levantou as propriedades qualitativas de *r* ótimo:
  - Análise com Dimensão VC.
  - Em problemas de baixa complexidade (# de respostas desejadas comparado ao número de amostras), desempenho da validação cruzada é insensível a *r*.
  - Um único *r* pode ser adequado para um conjunto grande de funções.

Protocolo de Treinamento

- Treinamento estocástico:
  - Seleção aleatória de amostras.
- Treinamento *Batch*
  - Época (Epoch): Apresentação simples de todos padrões de treinamento.
  - Pesos são atualizados apenas uma vez por época:

$$\Delta w_{ji}(n) = -\eta \frac{\partial E_{av}(n)}{\partial w_{ji}(n)} = -\frac{\eta}{N_{pad}} \sum_{n=1}^{Npad} e_j(n) \frac{\partial e_j(n)}{\partial w_{ji}(n)}$$

Protocolo de Treinamento

- Modo Seqüencial:
  - Atualiza-se pesos para cada amostra de treinamento.
  - Menos armazenamento, maior velocidade de convergência.
  - Risco: Ordem seqüencial pode levar a convergências para ótimos locais.
- Treinamento on-line:
  - Dados de treinamento abundantes.
  - Alto custo para armazenamento.

Protocolo de Treinamento

- Maximização do conteúdo de informação: Todo exemplo propiciar mais pesquisa no espaço de pesos:
  - Uso de um exemplo que resulta no maior erro (exemplo difícil) com respeito às iterações prévias.
  - Uso de um exemplo radicalmente deferente do demais.
  - Emprego de ordem aleatória para apresentação de exemplos para assegurar propriedade de pertinência a uma dada classe.
  - Ênfase em tratar mais amostras difíceis que fáceis, tendo cuidado com a distorção da distribuição dos exemplos e com presença de "outliers".
- Valores alvos devem ser tratáveis pela função de ativação, isto é, o valor do padrão desejado deve ser atingido pela função.
  - Quando isto não acontece, os pesos podem tender a infinito.

Algoritmo MLP-BP

- Inicialize aleatoriamente pesos e bias;
- Enquanto E_{av} for insatisfatório e houver disponibilidade computacional:
  - Para cada padrão de entrada:
    - Compute a soma ponderada dos nodos escondidos;
    - Compute a resposta dos nodos escondidos;
    - Compute a soma ponderada dos nodos escondidos;
    - Compute a resposta dos nodos escondidos;
    - Modifique os pesos que chegam à camada de saída;
    - Modifique os pesos que chegam a cada uma das camadas escondidas;
  - Fim-do-Para
- Fim-do-Enquanto

Referências

- Hassoun, M. H. (1995). *Fundamentals of Artificial Neural Networks*. Cambridge: The MIT Press.
- Haykin, S. O. (2009). *Neural Networks and Learning Machines*. McMaster University, Ontario Canada, 3rd ed.