Rede Perceptrons de Múltiplas Camadas - Algoritmo da Retropropagação (Backpropagation)

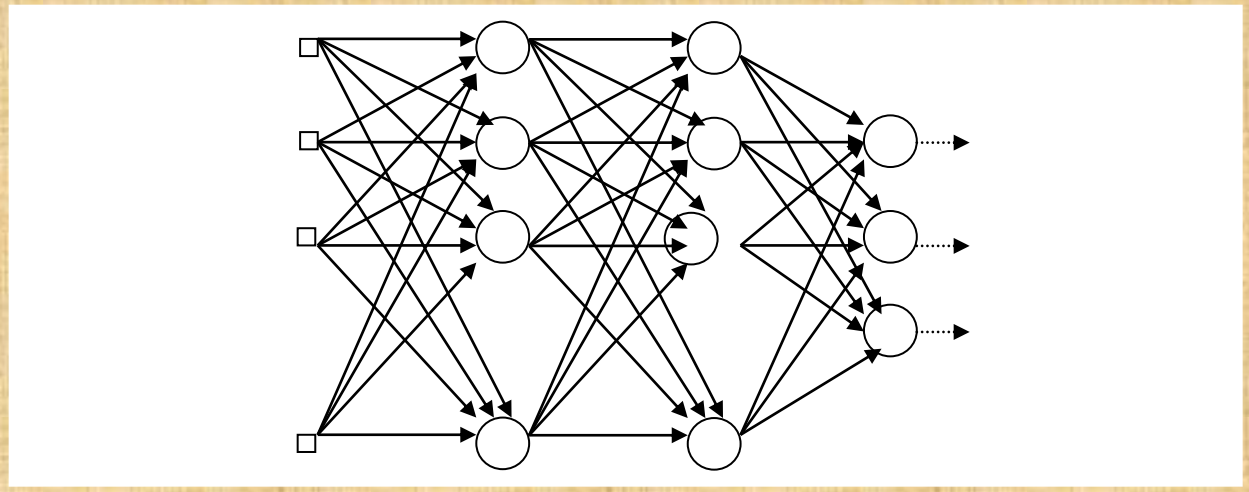
PROFESSOR ADRIÃO DUARTE D.NETO

DCA-CT-UFRN

Plano da apresentação

- **Rede Perceptrons de Múltiplas Camadas - MLP**
- **Derivação do Algoritmo Backpropagation**
- **Algoritmo Backpropagation**
- **Teoria da Regularização**
- **Aplicações do Perceptron de Múltiplas Camadas**

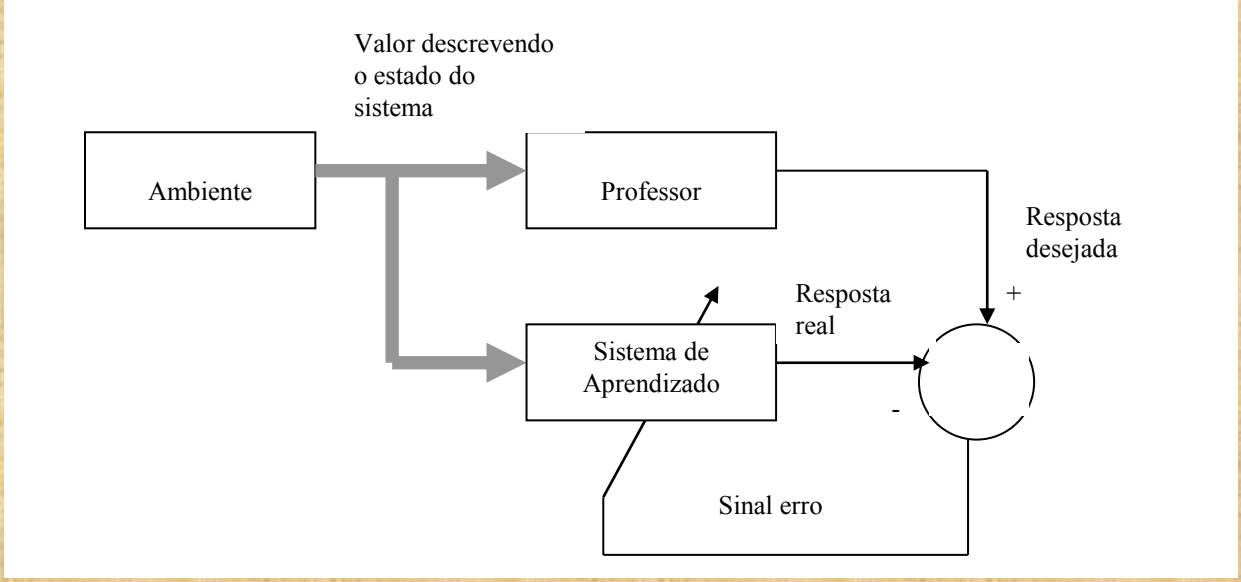
Perceptrons de Múltiplas Camadas



Mapeamento - $F : \mathbb{R} \qquad \mathbb{R}$

Perceptrons de Múltiplas Camadas

- Treinamento supervisionado



Derivação Ado Algoritmo da Retropropagação

Seja:

$$e_j(n) = d_j(n) - y_j(n)$$

o sinal erro para o neurônio j na camada de saída da rede.

- O funcional dos erros médios quadrados instantâneos é dado por: $\varepsilon(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n)$

- O funcional da média dos erros médios quadrados instantâneos é dado por:

$$\varepsilon_m = \frac{1}{2N} \sum_{n=1}^{N} \varepsilon(n)$$

Derivação do Algoritmo da Retropropagação

A função de ativação interna do neurônio j:

$$v_j(n) = \sum_{i=0}^{p} w_{ji}(n) y_i(n)$$

$$y_j(n) = \varphi(v_j(n))$$

Objetivo minimizar $\varepsilon(n)$ em função do ajuste dos ganhos sinápticos $w_{ji}(n)$.

Método do Gradiente:

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

$$w_{ji}(n+1) = w_{ji}(n) - \eta(n) \frac{\partial \varepsilon(n)}{\partial w_{ji}(n)}$$

Assim o gradiente local é dado por

$$\gamma_j(n) = \begin{cases} e_j(n)\varphi'(v_j(n)); j \in C \\ \varphi'(v_j(n))\sum_k \gamma_k(n) w_{kj}(n); j \notin C \end{cases}$$

Para o caso da função de ativação sigmoide.
O ajuste dos pesos sinápticos é dada por\;

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) = w_{ji}(n) + \eta(n)\gamma_j(n)y_i(n)$$

Uso do termo momento:

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

$$w_{ji}(n+1) = w_{ji}(n) + \alpha \Delta w_{ji}(n-1) + \eta(n)\gamma_j(n)y_i$$

onde, $0 \leq |\alpha| \leq 1$,

Algoritmo da Retropropagação

Modo de ajuste por lote "batch".

- Os ganhos são atualizados após a apresentação de todos os exemplos de treinamento,

$$\varepsilon_m = \frac{1}{2N} \sum_{n=1}^{N} \sum_{j \in C} e_j^2(n)$$

$$\Delta w_{ji}(n) = -\frac{\eta}{N} \sum_{n=1}^{N} e_j(n) \frac{\partial e_j(n)}{\partial w_{ji}(n)}$$

Algoritmo da Retropropagação

Resumo do Algoritmo da Retropropagação

1. Inicialização: Inicializar todos os ganhos e limiares da rede para valores aleatórios pequenos e uniformemente distribuídos (usar a função rand)
2. Apresentação dos exemplos de treinamento: Apresentar os exemplo de forma randonica para cada época (escolher aleatoriamente ou embaralhar os exemplos de cada época)
3. Cálculo de propagação direta:

Dado um exemplo, calcular as funções de ativação e a função de saída de cada neurônio.

$$v_j^{[l]}(n) = \sum_{i=0}^{p} w_{ji}^{[l]}(n) y_i^{[l-1]}(n)$$

$$y_j^{[l]}(n) = \varphi(v_j^{[l]}(n))$$

$$y_i^{[0]} = x_j(n)$$

Algoritmo da Retropropagação

4. Para última camada l=L, calcular os sinais erros e os gradientes locais

$$e_j^{[L]}(n) = d_j(n) - y_j^{[L]}(n)$$

$$\gamma_j^{[L]} = e_j^{[L]}(n)\varphi'(v_j(n))$$

5. Cálculo de Propagação reversa e cálculo do gradiente local nas camadas ocultas

$$\gamma_k^{[l+1]}(n) = e_k^{[l+1]}(n)\varphi_k'^{[l+1]}(n)$$

$$\gamma_j^{[l]}(n) = \varphi_j'^{[l]}(n)\sum_k \gamma_k^{[l+1]}(n) w_{kj}^{[l+1]}(n)$$

Algoritmo da Retropropagação

6. Atualização dos Ganhos e Limiares em cada camada

$$w^{[l]}_{ji}(n+1) = w^{[l]}_{ji}(n) + \Delta w^{[l]}_{ji}(n)$$

$$w^{[l]}_{ji}(n+1) = w^{[l]}_{ji}(n) + \alpha \Delta w^{[l]}_{ji}(n-1) + \eta(n)\gamma^{[l]}_{j}(n)y^{[l-1]}_{i}(n)$$

7. Retornar ao passo (2) até que todos os exemplos forem apresentados e $\varepsilon_m(w_{final}) < tol_2$

Perceptrons de Múltiplas Camadas

- Considerações Práticas
- Tratamento prévio das entradas da rede
  - Embaralhar o conjunto de treinamento
  - Aumentar a frequência dos padrões que produzem maior erro
  - Normalizar
  - Escalonar
  - Utilizar os padrões os mais descorrelacionados possível

Perceptron de Múltiplas Camadas

Algoritmo da Retropropagação

- **Generalização:**

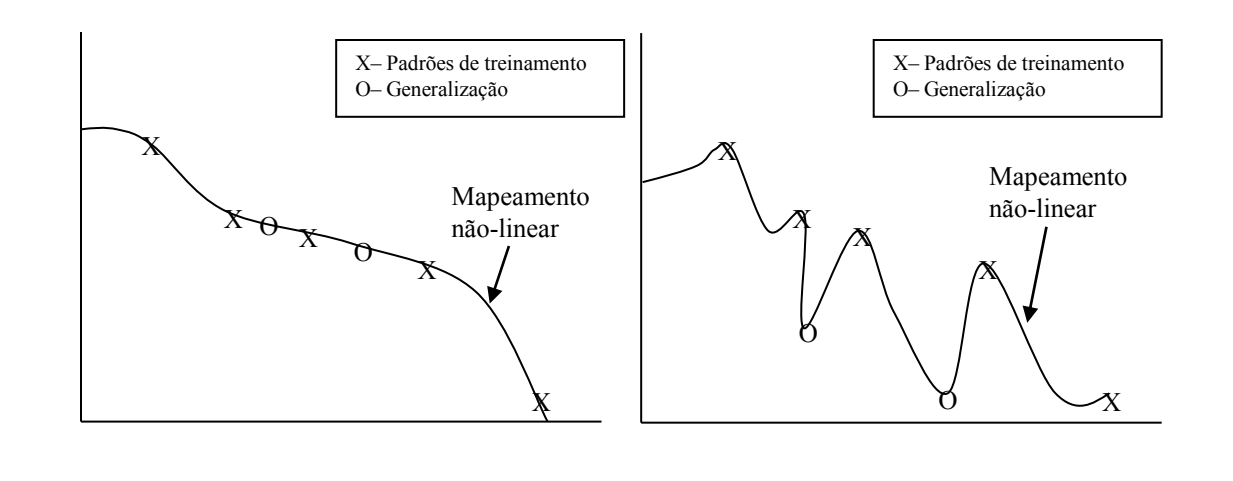
- A rede atua com uma função mapeando entrada/saída, para valores diferentes dos valores de treinamento.
- Problema de ajuste de dados
- Interpolação não linear dos dados de treinamento

A generalização é influenciada por:

. tamanho e eficiência do conjunto de treinamento
. arquitetura da rede
. complexidade do problema

Perceptrons de Múltiplas Camadas

- Generalização



  - Adequada seleção de exemplos de treinamento
  - Capacidade de evitar mínimos locais da função custo

Perceptrons de Múltiplas Camadas

- Validação Cruzada
- O conjunto de dados é dividido aleatoriamente em um conjunto de treinamento e em um conjunto teste.
- O conjunto de treinamento é dividido em dois subconjuntos disjuntos:
  - Subconjunto de estimação
  - Subconjunto de validação

Método de treinamento com parada antecipada: Treinar, parar após um número de apresentações, validar, reiniciar o treinamento, parar, validar...Repetir o processo até que verifique-se que o erro de validação volta a crescer.

Perceptrons de Múltiplas Camadas

- Validação Cruzada
- Validação Cruzada Múltipla (pouco dados para validar)
- O conjunto de dados de N exemplos é dividido em K subconjuntos. O rede é treinada com os subconjunto exceto um e que é usado para validação. O processo se repete até que todos os subconjunto sejam utilizados na validação.

Perceptrons de Múltiplas Camadas

Regularização da Complexidade

$$R(\mathbf{w}) = \varepsilon_S(\mathbf{w}) + \lambda \varepsilon_C(\mathbf{w})$$

Regularização L2

Regularização L1

Dropout

Teoria da Regularização

2.2.2.1 Decaimento dos pesos (*weight decay*)

Uma das formas mais simples e mais utilizadas para o funcional regularizador $E_c(\underline{w})$ é dada pelo quadrado da norma euclidiana do vetor de parâmetros $\underline{w}$, dado por:

$$E_c(\underline{w}) = \|\underline{w}\|^2 = \sum_{j=1}^{M} w_j^2 \tag{2.25}$$

Apesar deste funcional regularizador ser bastante utilizado na literatura, uma das suas limitações reside na sua inconsistência com relação às propriedades de escalonamento dos mapeamentos realizados pelos MLP's [158]. Entretanto, esta indesejável característica pode ser superada através da utilização de um outro regularizador, dado pela seguinte equação, para MLP's com uma única camada oculta:

$$E_c(\underline{w}) = \sum_{j=1}^{M_1} w_{1j}^2 + \sum_{j=1}^{M_2} w_{2j}^2 \tag{2.26}$$

Na equação (2.26), $w_i = \left[w_{i1}, w_{i2}, \ldots, w_{iM_i} \right]^t$ representa o conjunto de pesos sinápticos

Deep Feedforward Networks

L1 **Regularization:**

Formally, L^1 regularization on the model parameter $\boldsymbol{w}$ is defined as:

$$\Omega(\boldsymbol{\theta}) = ||\boldsymbol{w}||_1 = \sum_i |w_i|, \tag{7.18}$$

that is, as the sum of absolute values of the individual parameters.[2] We will now discuss the effect of L^1 regularization on the simple linear regression model, with no bias parameter, that we studied in our analysis of L^2 regularization. In particular, we are interested in delineating the differences between L^1 and L^2 forms

[2] As with L^2 regularization, we could regularize the parameters towards a value that is not zero, but instead towards some parameter value $\boldsymbol{w}^{(o)}$. In that case the L^1 regularization would introduce the term $\Omega(\boldsymbol{\theta}) = ||\boldsymbol{w} - \boldsymbol{w}^{(o)}||_1 = \sum_i |w_i - w_i^{(o)}|$.

Teoria da Regularização

- **Dropout:** provides a computationally inexpensive but powerful method of regularizing a broad family of models. Specifically, dropout trains the ensemble consisting of all sub-networks that can be formed by removing non-output units from an underlying base network.

Algoritmo da Retropropagação

Algoritmo da Retropropagação

-

- É atualmente o mais utilizados dos algoritmos supervisionados.
- Baixa complexidade computacional
- Fácil ajuste dos ganhos

Convergência: Método do gradiente de primeira ordem de natureza estocástica. Alternativa, usar outros métodos (Newton, Marquand-Levemberg, Gradiente conjugado, etc.)

Algoritmo da Retropropagação

Método do Gradiente Conjugado

$$\mathbf{w}(n+1)=\mathbf{w}(n)+\eta(n)\mathbf{p}(n)$$
$$\mathbf{p}(0)=-\mathbf{g}(0)$$
$$\mathbf{g}(\mathbf{n})=\mathbf{gradiente}$$
$$\mathbf{p}(n+1)=-\mathbf{g}(n+1)+\beta(n)\mathbf{p}(n)$$
$$\beta(n)=\frac{\mathbf{g}^t(n+1)\mathbf{g}(n+1)}{\mathbf{g}^t(n)\mathbf{g}(n)}$$
$$\eta(n)=\min\{\varepsilon(\mathbf{w}(n)+\eta(n)\mathbf{p}(n))\}$$

Algoritmo da Retropropagação

Método de Newton

$$\Delta\varepsilon_m(\mathbf{w}) = \varepsilon(\mathbf{w}+\Delta\mathbf{w}) - \varepsilon(\mathbf{w})$$

$$\Delta\varepsilon_m(\mathbf{w}) \cong \mathbf{g}^t\Delta\mathbf{w} + \frac{1}{2}\Delta\mathbf{w}^t\mathbf{H}\Delta\mathbf{w}$$

$$\mathbf{H} = \frac{\partial\varepsilon_m(\mathbf{w})}{\partial\mathbf{w}^2}$$

$$\mathbf{g} + \mathbf{H}\Delta\mathbf{w} = 0$$

$$\Delta\mathbf{w} = -\mathbf{H}^{-1}\mathbf{g}$$

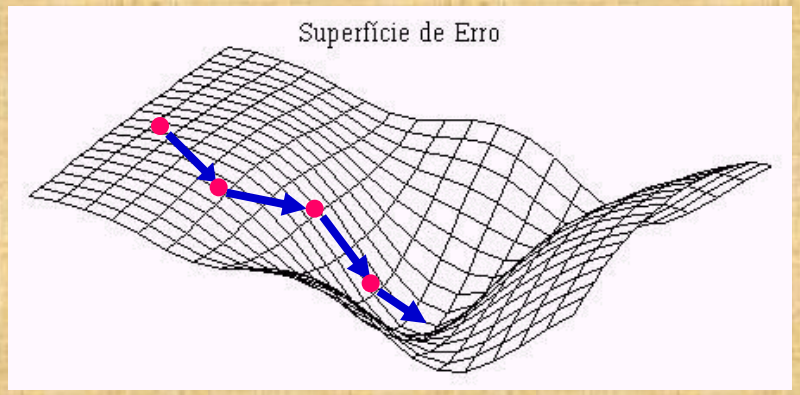
$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mathbf{H}^{-1}(n)\mathbf{g}(n)$$

Mínimo Local: O algoritmo não garante um mínimo global, portanto pode convergir para um mínimo local, Alternativa: Algoritmos genéticos.

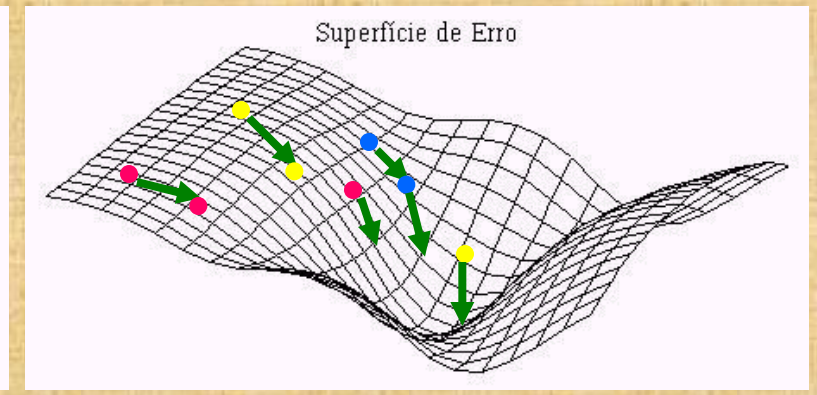
Algoritmos paralelos em perceptrons multicamadas com treinamento por retropropagação

- Matrizes de ganhos sinápticos próprias para cada tarefa

Paralelo Clássico



Paralelo Competitivo



Perceptron de Múltiplas Camadas

Conexionismo:

a) – Metáforas de rede neurais biológicas
b) - Conexão direta somente com os neurônios fisicamente co-
nectados.
c) - Robustez
d) – Implementação em paralelo

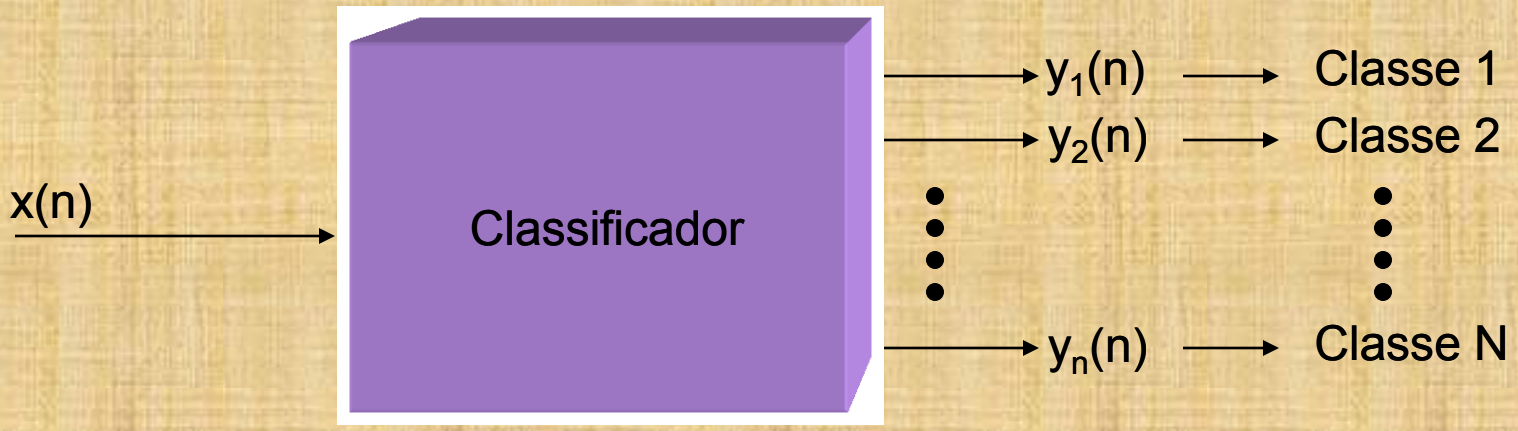
Camadas Ocultas: atuam como detetores de características

Aproximador Universal:

$$F(x,W)=\varphi\left(\sum_j w_{sj}\varphi\left(\sum_k w_{jk}\varphi\left(\cdots\varphi\left(\sum_i w_{li}\right)\cdots\right)\right)\right)$$

Perceptron de Múltiplas Camadas

- Classificação de Padrões



Classificação de Padrões

- Seja $\mathbf{x}_j$ um vetor aleatório escolhido em uma distribuição de dados e a ser classificado em uma das m classes distintas.
- Cada classe é associada através do vetor de saída $\mathbf{y_j}=\mathbf{F}(\mathbf{x_j})$
- Procedendo um treinamento supervisionado, isto selecionado um conjunto de treinamento com N exemplos e associando a cada entrada o vetor resposta desejada $\mathbf{d}(n)=[0,0,\ldots,1,\ldots 0]$

$$d_{k,j}=\begin{cases}1 & \mathbf{x}_j \in C_k \\ 0 & \mathbf{x}_j \notin C_k\end{cases}$$

Classificação de Padrões

- Funções de Ativações na Saída:
- Função Sigmoíde para Classificação Binária
- Função Softmax para Classificação M-ária

Classificação de Padrões

- Definindo a função custo a ser minimizada como

$$\varepsilon_m = \frac{1}{2N} \sum_{j=1}^{N} \|\mathbf{d}_j - \mathbf{F}(\mathbf{x}_j)\|^2$$

- Após o treinamento usar o critério da maior saída, isto é,

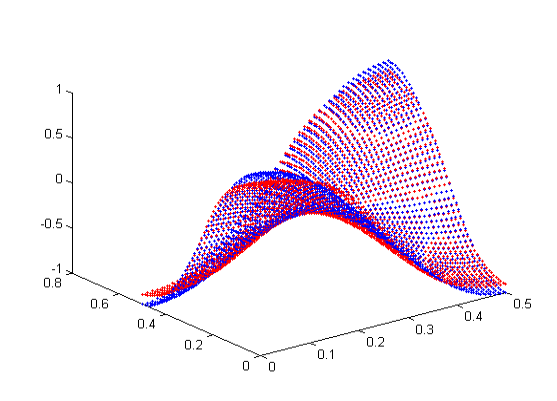
$$\mathbf{x}_j \in C_k \; se \quad y_k(\mathbf{x}_j) > y_i(\mathbf{x}_j) \quad \forall i \neq k$$

Aplicação desenvolvida e resultados

- Resultados
  - Aproximação de funções

$$f(x_1, x_2) = \cos(2\pi x_1)\cos(2\pi x_2) \qquad 0 <= x_1 <= 0{,}5 \qquad 0 <= x_2 <= 0{,}5$$

    - Topologia utilizada foi 2:10:1 (incluir ganhos paralelos)



Aplicações

- Identificação de Sistemas
- Controle
- Reconhecimento de Voz
- Visão computacional
- Reconhecimento de caracteres escrito a mão
- Detecção e Classificação de falhas
- Diagnósticos médicos
- Etc..

Conclusões

- A rede perceptron de múltiplas camadas é a mais tradicional das redes neurais
- Possui a propriedade de aproximador universal
- Problemas de overfitting para arquiteturas mais complexas
- Base para redes deep learning