## Algoritmos de Aprendizado

- **Regra de Hebb**
- **Perceptron**
- **Delta Rule (Least Mean Square)**
- **Back Propagation**
- **Hopfield**
- **Competitive Learning**
- **Radial Basis Function (RBFs)**

## Radial Basis Functions

- Probabilistic Neural Networks
- RBFs
- Generalized Regression Neural Networks

# Radial Basis Functions (RBFs)

- As redes RBFs fazem parte, juntamente com as redes PNN, de uma classe de técnicas denominadas ***Gaussian Potential Functions*** para ***Classificação*** e ***Aproximação de Funções***.

# Radial Basis Functions (RBFs)

- **<u>Vantagens Adicionais ao BP:</u>**
  - Aprendizado rápido
    - Não começa de pesos aleatórios
  - Treinamento Incremental
  - Sem problemas tipo Paralisia da Rede e Mínimo Local

# Radial Basis Functions (RBFs)

- **<u>Desvantagem Básica:</u>**
  - Após o treinamento, as redes RBFs são ***mais lentas*** para efetuar a ***recuperação da informação (Recall)***

# Radial Basis Functions (RBFs)

- As redes RBFs são ***aproximadores universais***, isto é, dado um **número suficiente de neurônios na camada escondida**, as redes RBFs podem **aproximar qualquer função contínua** com precisão arbitrária

# Radial Basis Functions (RBFs)

- **Idéia básica:**
  - Supondo que os vetores de *entrada* e *pesos* definam pontos no espaço N-dimensional, o objetivo é ter uma *resposta do neurônio que diminua rapidamente conforme esses pontos se distanciem*

# Radial Basis Functions (RBFs)

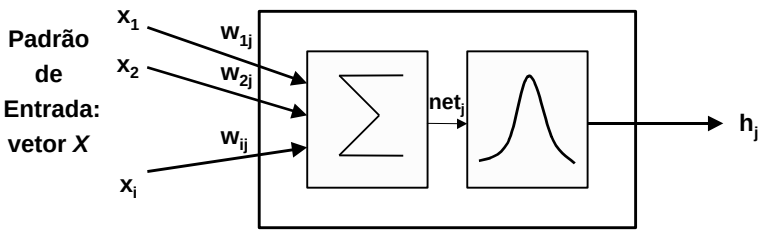
- **Idéia básica:**
  - Supondo que os vetores de *entrada* e *pesos* definam pontos no espaço N-dimensional, o objetivo é ter uma *resposta do neurônio que diminua rapidamente conforme esses pontos se distanciem* ➔funções gaussianas ➔ RBFs

# Radial Basis Functions (RBFs)

- **Idéia básica:**
  - Supondo que os vetores de *entrada* e *pesos* definam pontos no espaço N-dimensional, o objetivo é ter uma *resposta do neurônio que diminua rapidamente conforme esses pontos se distanciem* ➔funções gaussianas ➔ RBFs
  - O *conjunto de neurônios escondidos* é projetado de forma que as suas *respostas cubram todas as regiões* significativas do espaço de vetores de entrada

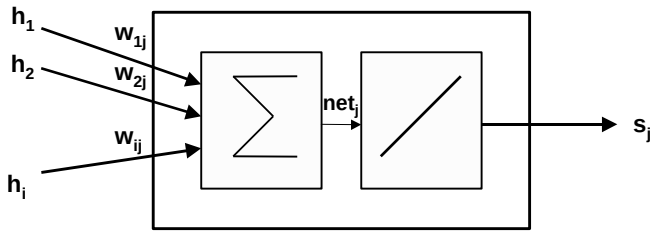
# RBFs

- ***Elementos Processadores:***
  - Camada Escondida



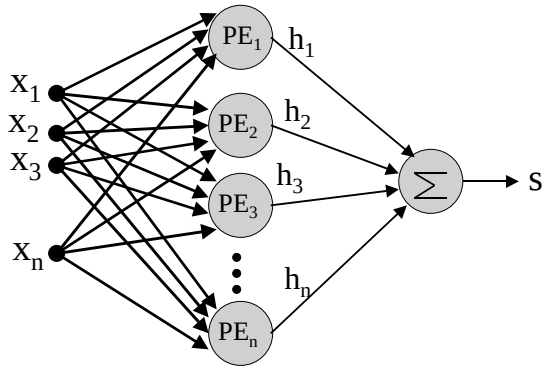
# RBFs

- ***Elementos Processadores:***
  - Camada de Saída




# RBFs

- ***Topologia com uma única saída:***



# RBFs

- ***Características Básicas:***
  - Regra de Propagação ➔ $net_j = S\ x_i.w_{ij} = X.W_j$
    *(para vetores normalizados)*
  - Função de Ativação ➔ ***Gaussiana*** *(camada escondida)*
    ***e Linear*** *(camada saída)*

    $h_i = \exp[(net_i - 1)/\sigma^2]$
    $h_i = \exp[-D_i^2/2]$ onde $D_i^2 = (x - u)^t (x - u)$
    $s = net$

    Vetor de entrada ↗ ↖ Vetor de treinamento
  - Topologia ➔ ***Duas camada de processadores.***
  - Algoritmo de Aprendizado ➔ **Supervisionado**
  - Valores de Entrada/Saída ➔ ***Binários/Contínuos***

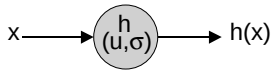
# RBFs

***<u>Observações</u>:***

- Para as ***PNNs*** e ***GRNNs*** (a serem vistas), os ***pesos são fixos***.
- Em outros casos, somente os ***pesos de saída*** são modificados ➔ ***treinamento de uma rede de uma camada linear***
- Caso geral, ***todos os pesos*** são ajustados, além da forma da função gaussiana.

# RBFs

- ***Operação da Rede:***
  - apenas uma entrada x

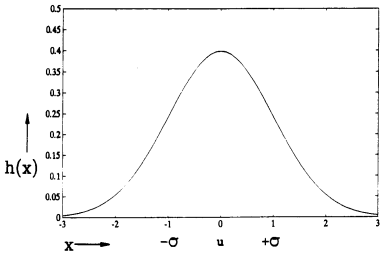



$$h = \exp\left[(x - u)^2 / 2\sigma^2\right]$$

**u = w = valor de treinamento**
**= média**

**σ = desvio padrão**

***Função de Ativação do Neurônio***





# RBFs


- ***Operação da Rede:*** apenas uma entrada x
  - ***Quando* x = u h(x) = 1.0**

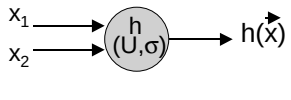
    **Portanto, *u* determina o valor de *x* que produz *saída máxima*. A resposta para outros valores de *x cai rapidamente* conforme *x se afasta* de *u*.**
  - **A *saída do neurônio* tem resposta significativa para a entrada *x* somente sobre *uma faixa de x***

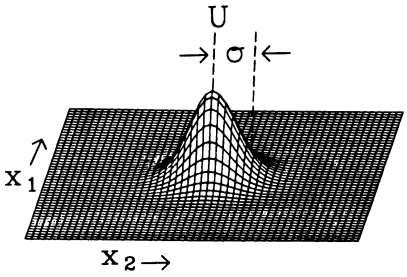
    **campo receptivo (σ)**

# RBFs

- ***Operação da Rede:*** duas entradas $x_1$ e $x_2$

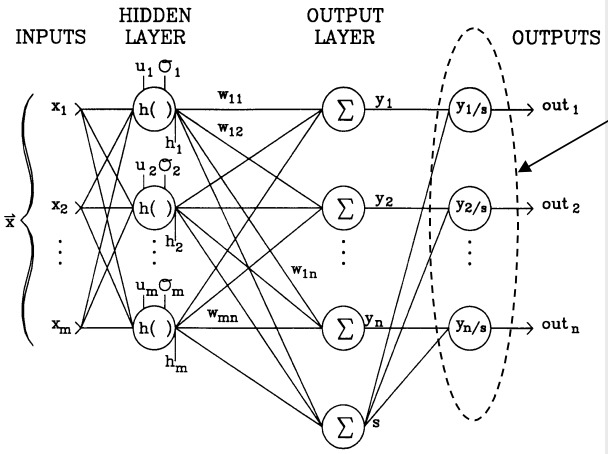
***Função de Ativação do Neurônio Bi-dimensional***




# RBFs

- ***Topologia com múltiplas saídas:***




***Função de Normalização***

divide a saída de cada processador da camada de saída pela soma de todas as saídas dos processadores da camada escondida

$$out_j = \frac{\Sigma^m h_i w_{ij}}{\Sigma^m h_i}$$

# RBFs

- ***Topologia com múltiplas saídas:***
  - Esta é a **forma geral** que, com valores específicos de pesos, representa a GRNN - ***Generalized Regression Neural Network***

# RBFs - Operação da Rede

- ***Treinamento:***
  - ***os parâmetros ajustáveis da rede (*** $u_i$, $\sigma_i$ ***e*** **W*****) são estabelecidos de forma a minimizar o erro médio entre a saída da rede e a saída desejada***

## RBFs - Operação da Rede

- ***Treinamento:***
  - ***os parâmetros ajustáveis da rede ($u_i$, $\sigma_i$ e W) são estabelecidos de forma a minimizar o erro médio entre a saída da rede e a saída desejada***
- ***Referência (Recall):***
  - ***os valores de entrada são apresentados à rede e os vetores de saída são produzidos.***

## RBFs - Operação da Rede

- ***Treinamento em dois estágios:***

  ***designar valores para o centro $u_i$ (vetor de pesos) e o desvio padrão $\sigma_i$ para cada neurônio da camada escondida***

  ***treinar a matriz de pesos dos neurônios da camada de saída método supervisionado***

# RBFs - Treinamento

- ***Fase I: Localização dos Centros $u_i$***

***Questão crítica, com várias alternativas para a sua determinação***

**método mais simples** — **um centro para cada vetor de entrada do conjunto de treinamento**

⬇

**como os vetores de treinamento tendem a possuir grupos, esse método resulta em mais processadores que o necessário**

⬇

**método mais complexo** — **determina-se o centro de cada grupo de vetores, estabelecendo um processador para cada grupo (métodos de clustering)**

# RBFs - Treinamento

- ***Fase I: Determinação de $\sigma_I$***

***O diâmetro do Campo Receptivo pode ter um grande efeito na precisão do sistema***

⬇

***Heurística:***

***1 - Para cada processador da camada escondida, determine a distância RMS entre o seu centro*** $u_i$ ***e o de seus*** **N** ***vizinhos***

***2 - Atribua este valor para*** $\sigma_i$

# RBFs - Treinamento

- ***Fase II*: *Treinamento dos pesos da Camada de Saída - W***

***Treinamento supervisionado*** **(vetores $x_i \Rightarrow t_i$)**

⬇

***1 - Aplica-se um vetor de entrada $x_i$ do conjunto de treinamento***

***2 - Calcula-se as saídas dos processadores da camada escondida*** **$\Rightarrow$ vetor h**

***3 - Computa-se o vetor de saída y e compara-se com $t_i$ $\Rightarrow$ ajusta-se W na direção de reduzir a diferença*** $\Delta w_{ij} = \eta . h_i . (t_j - y_j)$

***4 - Repete-se 1 a 3 para cada vetor de treinamento***

***5 - Repete-se 1 a 4 até que o erro seja pequeno***

# RBFs - Treinamento

- ***Fase II*: *Treinamento dos pesos da Camada de Saída - W***

***Observação:*** ***Devido ao Campo Receptivo limitado de cada processador da camada escondida, a maioria das saídas será próxima de zero***

# RBFs - Treinamento

- ***Fase II: Treinamento dos pesos da Camada de Saída - W***

***Observação:*** ***Devido ao Campo Receptivo limitado de cada processador da camada escondida, a maioria das saídas será próxima de zero*** $\Rightarrow h_i \approx 0$ **para um certo x** $\Rightarrow \Delta w_{ij} \approx 0 \Rightarrow$ **pode-se concentrar apenas nos pesos dos** $PE_i$ **com** $h_i > 0$

# Generalized Regression Neural Networks - GRNN

- ***Características: (Specht 91)***
  - Não possui treinamento iterativo;
  - Aproxima qualquer função entre entrada e saída (a partir dos dados de treinamento);
  - Consistente conforme o conjunto de treinamento cresce, o erro se aproxima de zero
  - Método baseado na ***Teoria de Regressão Não-Linear***

# Generalized Regression Neural Networks - GRNN

- ***Regressão:***
  - Por definição, a ***regressão*** de uma variável dependente ***y*** sobre uma variável independente ***x estima o valor mais provável de y***, dado ***x*** e um conjunto de treinamento, que pode conter ruído.

# Generalized Regression Neural Networks - GRNN

- O método de Regressão produz um valor estimado de ***y*** que minimiza o ***Erro Quadrático Médio***

$$E[y|x] = \frac{\int_{-\infty}^{\infty} y\, f(x,y)\, dy}{\int_{-\infty}^{\infty} f(x,y)\, dy}$$

***Valor esperado de saída, dado o vetor de entrada x***

***Função Densidade de Probabilidade de x e y GRNN é um método para se estimar f(x,y) a partir dos dados***

# Generalized Regression Neural Networks - GRNN

- Specht demonstrou que $y_i$ é estimado otimamente da seguinte forma:

$$y_j = \frac{\sum_{i=1}^{n} y_{ij} h_i}{\sum_{i=1}^{n} h_i}$$

***Saída desejada para o processador j com o vetor de entrada i***

$$h_i = \exp[-(x-u_i)^t (x-u_i)/2\sigma^2]$$



# Generalized Regression Neural Networks - GRNN

- Specht demonstrou que $y_i$ é estimado otimamente da seguinte forma:

$$y_j = \frac{\sum_{i=1}^{n} y_{ij} h_i}{\sum_{i=1}^{n} h_i}$$

***Saída desejada para o processador j com o vetor de entrada i***

$$h_i = \exp[-(x-u_i)^t (x-u_i)/2\sigma^2]$$

***Fórmula idêntica à RBF com normalização, onde $y_{ij} = w_{ij}$***
***Não existe treinamento iterativo***

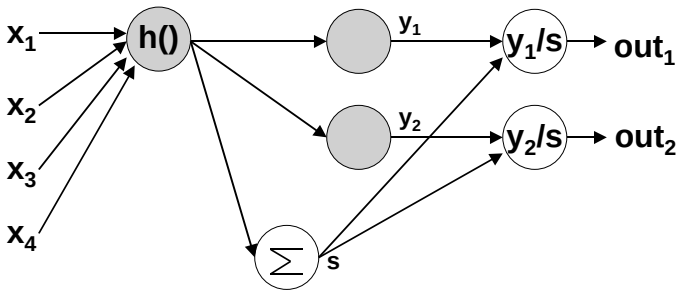
# Generalized Regression Neural Networks - GRNN

- Exemplo1: x = [1 2 5 3]
  y = [ 3 2 ]



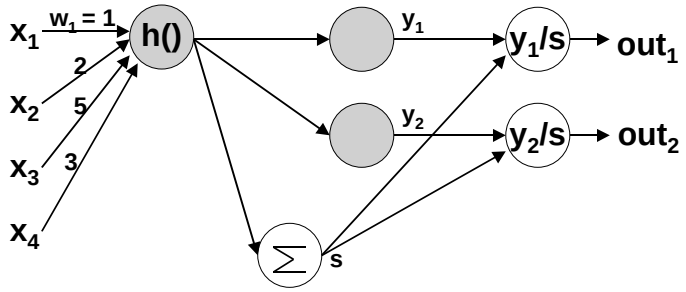
# Generalized Regression Neural Networks - GRNN

- Exemplo1: x = [1 2 5 3]
  y = [ 3 2 ]



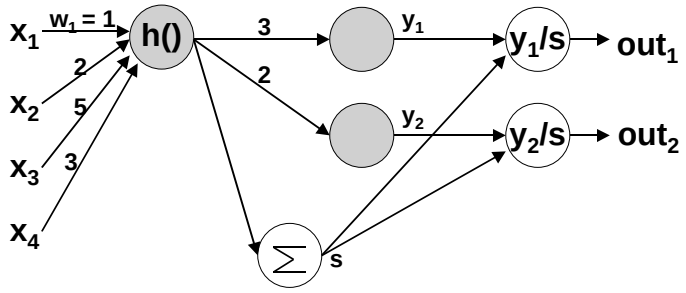

# Generalized Regression Neural Networks - GRNN

- Exemplo1: x = [1 2 5 3]
  y = [ 3 2 ]



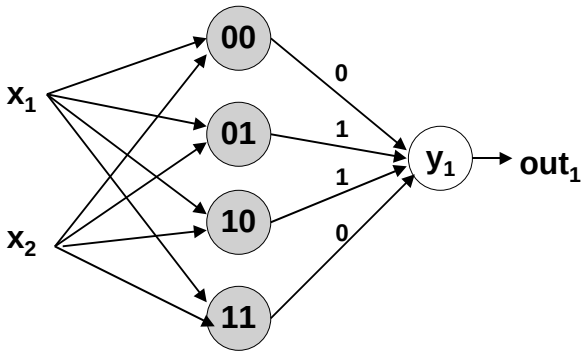

# Generalized Regression Neural Networks - GRNN

- Exemplo1: x = [1 2 5 3]
  y = [ 3 2 ]



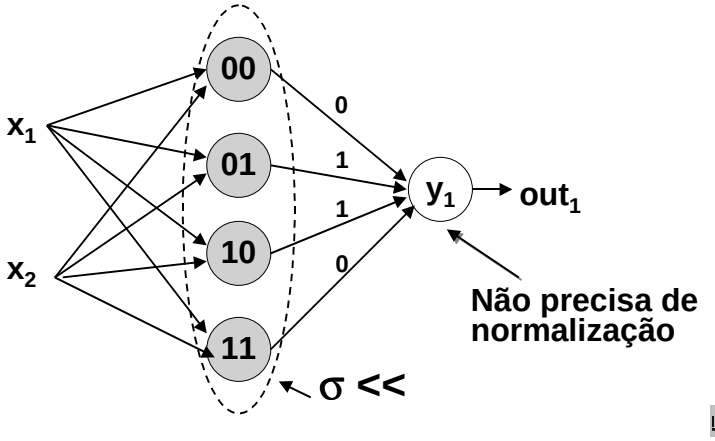
# Generalized Regression Neural Networks - GRNN

- Exemplo2: OU-EXCLUSIVO




# Generalized Regression Neural Networks - GRNN

- Exemplo2: OU-EXCLUSIVO



## Relação entre GRNN e PNN

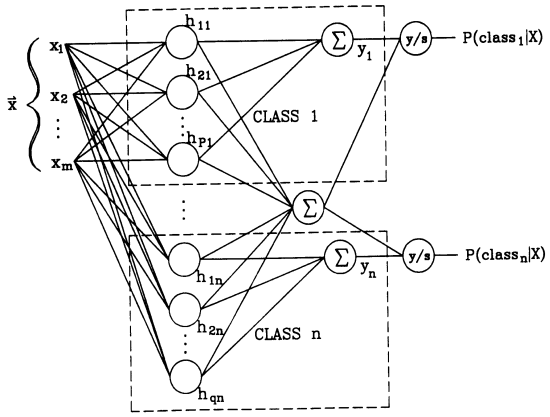
- Como a GRNN pode aproximar qualquer função, pode também ser uma rede ***Classificadora***
- Versão *normalizada da PNN* onde cada *saída* representa a *probabilidade de ser aquela classe*
- Como o classificador possui vetores de saída binários os pesos da camada de saída terão valores 0 e 1

## Relação entre GRNN e PNN

***Construção da Rede***:

– Cada PE da camada escondida representa um vetor de entrada do conjunto de treinamento
– Atribua o valor 1 ao peso indo para o PE de saída daquela classe
– Atribua o valor 0 a todos os outros pesos

# Relação entre GRNN e PNN




Adicionando a camada de decisão da PNN, as redes ficam equivalentes

# Relação entre GRNN e PNN

Observação:

- Apesar da PNN derivar dos Classificadores Bayesianos e a GRNN da Teoria de Regressão, demonstra-se que o método de Regressão é uma estimativa ótima no sentido do Erro Médio Quadrático (EMQ)
- Além disso, demonstra-se que qualquer estimador que seja ótimo no sentido do EMQ aproxima-se (com um número suficiente de dados de treinamento) de um Classificador Bayesiano.
- Portanto: GRNN aproxima-se de um Classificador Bayesiano $\Rightarrow$ PNN

# RBFs

- ***Conclusões:***
  - Técnicas ***RBFs*** são métodos poderosos, com grande aplicabilidade
  - Treinamento rápido
    - acelera o desenvolvimento e a avaliação do problema
  - Aprendizado Incremental

# RBFs

- ***Conclusões:***
  - Lento no Processo de Recuperação

A informação é localizada, isto é, os pesos são efetivos sobre uma pequena porção do espaço de entrada

O Back Propagation é totalmente distribuído

$\Rightarrow$ mais compacto

# RBFs

- ***Conclusões:***
  - Lento no Processo de Recuperação

A informação é localizada, isto é, os pesos são efetivos sobre uma pequena porção do espaço de entrada

O Back Propagation é totalmente distribuído

$\Rightarrow$ mais compacto

  - Problema na determinação do número de vetores de treinamento e o melhor $\sigma$

Deve ser adequado para cobrir o espaço de entrada com densidade suficiente para alcançar a precisão desejada

Deve ser grande o suficiente para produzir a generalização adequada

# RBFs

- ***Conclusões:***
  - Lento no Processo de Recuperação

A informação é localizada, isto é, os pesos são efetivos sobre uma pequena porção do espaço de entrada

O Back Propagation é totalmente distribuído

$\Rightarrow$ mais compacto

  - Problema na determinação do número de vetores de treinamento e o melhor $\sigma$

Deve ser adequado para cobrir o espaço de entrada com densidade suficiente para alcançar a precisão desejada

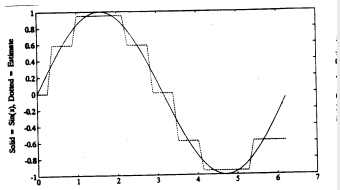
Deve ser grande o suficiente para produzir a generalização adequada

$\Rightarrow$ experimentos!

# RBFs

- ***Exemplo:*** Aproximação da Função Seno

  **Caso 1**: com 10 processadores na camada escondida

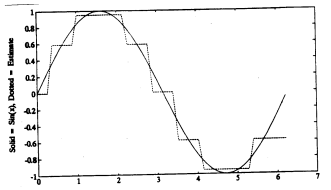



**σ = 0.01**

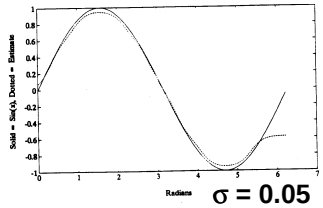
# RBFs

- ***Exemplo:*** Aproximação da Função Seno

  **Caso 1**: com 10 processadores na camada escondida




**σ = 0.01**

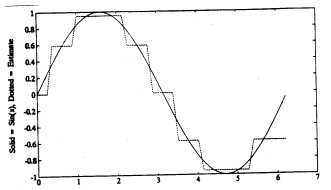



**σ = 0.05**

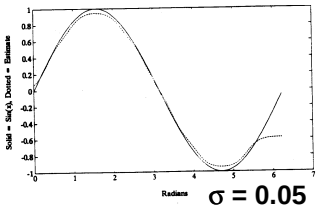
# RBFs

- ***Exemplo:*** Aproximação da Função Seno

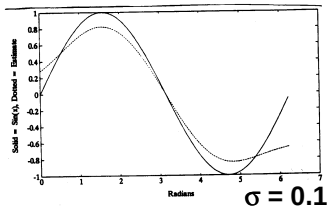
  **Caso 1**: com 10 processadores na camada escondida




σ = 0.01




σ = 0.05

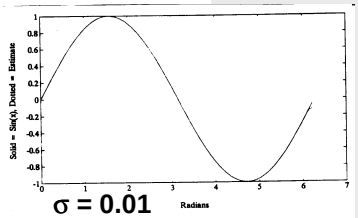



σ = 0.1

# RBFs

- ***Exemplo:*** Aproximação da Função Seno

  **Caso 2**: com 50 processadores na camada escondida

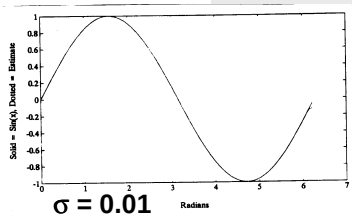



σ = 0.01

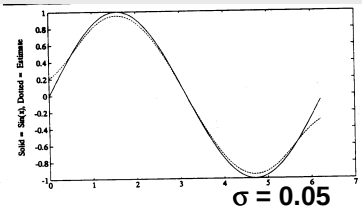
# RBFs

- ***Exemplo:*** Aproximação da Função Seno

  **Caso 2**: com 50 processadores na camada escondida




σ = 0.01



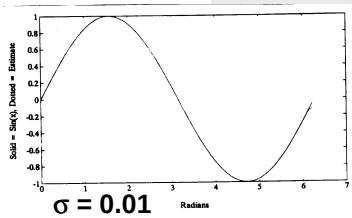

σ = 0.05



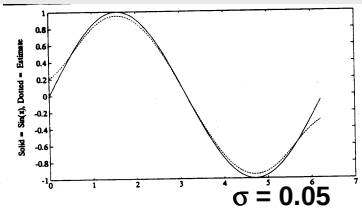
# RBFs

- ***Exemplo:*** Aproximação da Função Seno

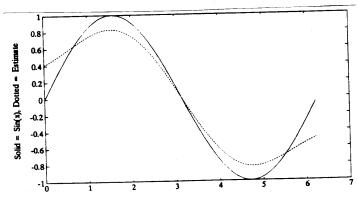
  **Caso 2**: com 50 processadores na camada escondida




σ = 0.01




σ = 0.05




σ = 0.1

# RBFs

- Em geral, conforme a ***densidade de padrões de entrada aumenta***, um valor ***menor*** de $\sigma$ deve ser usado!