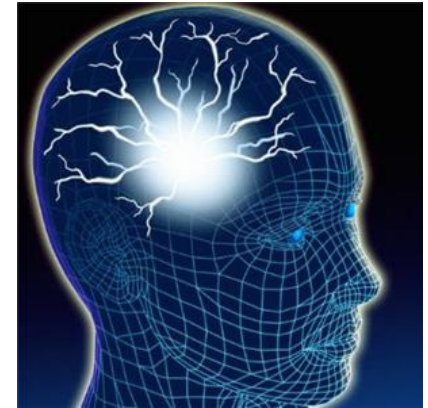


LSi



REDES NEURAIS

DECOM

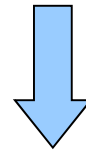
Rogério Gomes e Paulo Almeida

Redes Neurais: taxonomia

2

- ❑ O desenvolvimento da redes neurais se apoia na crença do seguinte paradigma para se conseguir produzir máquinas “inteligentes” :

Conexionismo



A criação de máquinas inteligentes é possível a partir da reprodução e interconexão dos elementos de processamento do cérebro dos animais.

Neurônios Biológicos e Modelos Artificiais

3

- ❑ O cérebro humano consiste de aproximadamente 10^{11} elementos de processamento chamados neurônios. Eles se comunicam através de uma rede de conexão formada por axônios e sinapses (cada neurônio possui em média 10^4 sinapses). A comunicação se processa essencialmente por meio de impulsos elétricos e reações químicas.

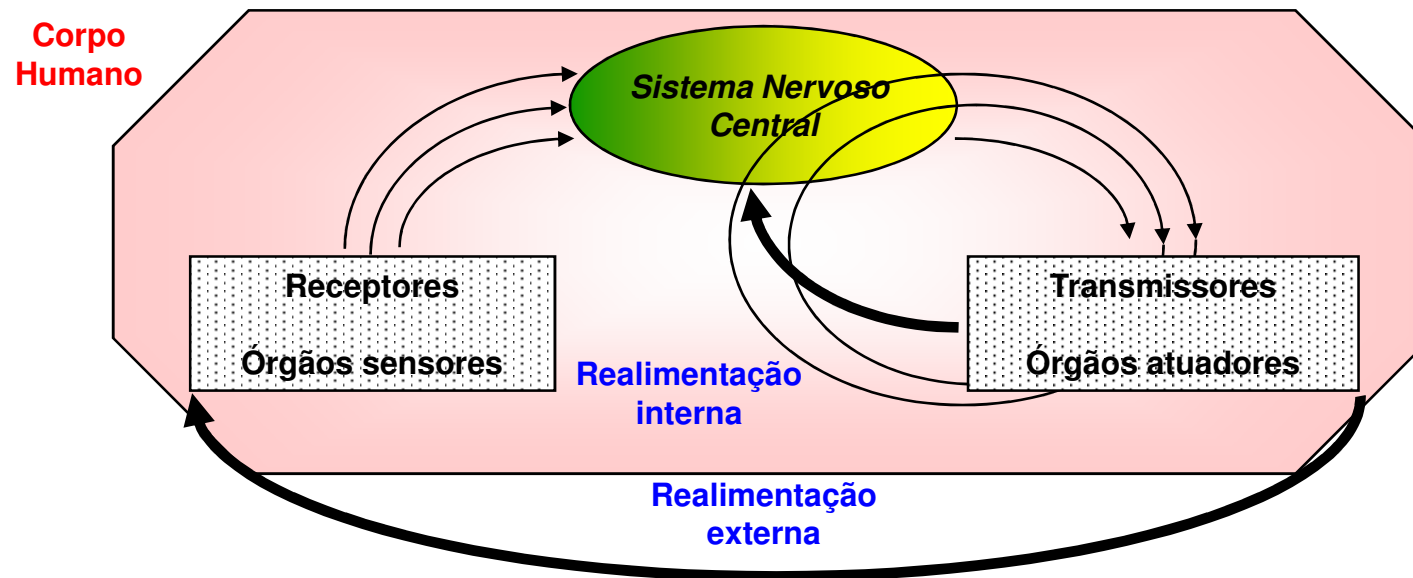
Neurônios Biológicos e Modelos Artificiais

4

- ❑ A figura a seguir mostra de forma esquemática o funcionamento geral do sistema nervoso humano, como um sistema de controle em três estágios: receptores, redes neurais de processamento e transmissores.

Neurônios Biológicos e Modelos Artificiais

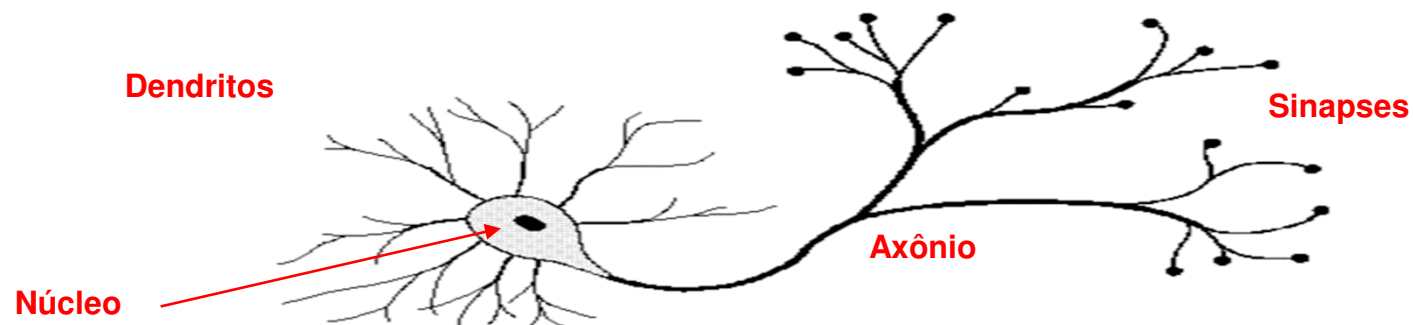
5



Neurônios Biológicos e Modelos Artificiais

6

- ❑ O elemento básico do sistema nervoso é o neurônio biológico. Abaixo é mostrada uma representação dos seus principais constituintes:



O Neurônio Artificial:

Modelos Matemáticos de Neurônios

7

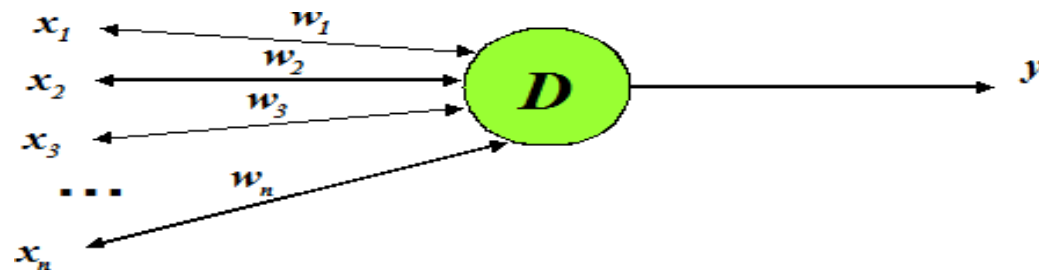
- ❑ Um neurônio artificial é representado por um modelo matemático simplificado do processamento existente em um neurônio biológico.
- ❑ O Neurônio de McCulloch e Pitts (ou Modelo MCP).

O Neurônio Artificial:

Modelos Matemáticos de Neurônios

8

- A primeira definição formal de um modelo de representação do neurônio biológico foi proposta em 1943, e é empregada até hoje em muitas aplicações. Seu diagrama é mostrado abaixo:



O Neurônio Artificial: Modelos Matemáticos de Neurônios

9

- ❑ As entradas x_i no Modelo MCP são binárias ($[0,1]$ ou $[-1,1]$), dependendo da presença ou ausência de um impulso de entrada no instante de tempo k , e os valores w_i são pesos multiplicativos conectando a entrada i ao núcleo do neurônio.

O Neurônio Artificial: Modelos Matemáticos de Neurônios

10

- ❑ A regra de disparo para este modelo é descrita a seguir.

$$y(k) = \begin{cases} 1, & \text{se } \sum_{i=1}^n w_i \cdot x_i(k) \geq D \\ 0, & \text{se } \sum_{i=1}^n w_i \cdot x_i(k) < D \end{cases}$$

- ❑ Na equação acima, D é um valor de limiar ou de disparo, a partir do qual o neurônio estará ativo e poderá contribuir para o disparo de outros neurônios conectados a ele.

O Neurônio Artificial: Modelos Matemáticos de Neurônios

11

- ❑ O Modelo MCP é muito importante pelo seu significado histórico, mas ele faz uso de simplificações muito drásticas que limitam a sua aplicação para problemas de maior complexidade. O emprego de entradas e saídas binárias é um exemplo de simplificação que limita a sua capacidade.
- ❑ Redes neurais artificiais (RNA) empregam uma grande variedade de modelos de neurônios diferentes, via de regra mais complexos que o modelo MCP.

O Neurônio Artificial: Modelos Matemáticos de Neurônios

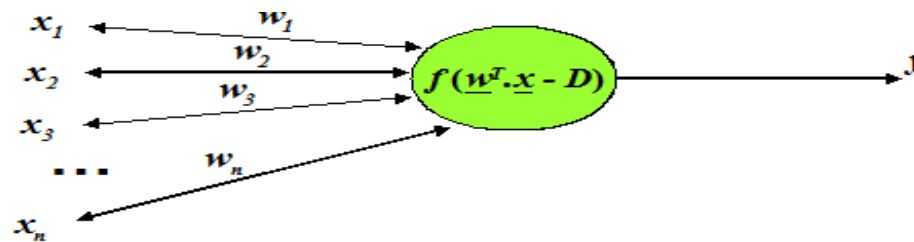
12

- ❑ Mas a grande maioria se baseia na mesma ideia principal de um elemento de processamento que possui:
 - ❑ Múltiplas entradas com conexões sinápticas representadas por pesos numéricos;
 - ❑ Núcleo do neurônio executando algum tipo de transformação nas entradas ponderadas;
 - ❑ Saída única.

O Neurônio Artificial: Modelos Matemáticos de Neurônios

13

- Um Modelo Geral de neurônio artificial é mostrado abaixo.



O Neurônio Artificial: Modelos Matemáticos de Neurônios

14

- ❑ O processamento que ocorre neste neurônio então se torna, de forma geral:

$$y(k) = f(w^T \cdot x - D) = f\left(\sum_{i=1}^n w_i \cdot x_i - D\right) = f(\text{net})$$

onde $w = [w_1 \quad w_2 \quad \dots \quad w_n]^T$ e $x = [x_1 \quad x_2 \quad \dots \quad x_n]^T$

- ❑ Todos os vetores definidos aqui são vetores-coluna.
O operador “T” denota a transposição.

O Neurônio Artificial: Modelos Matemáticos de Neurônios

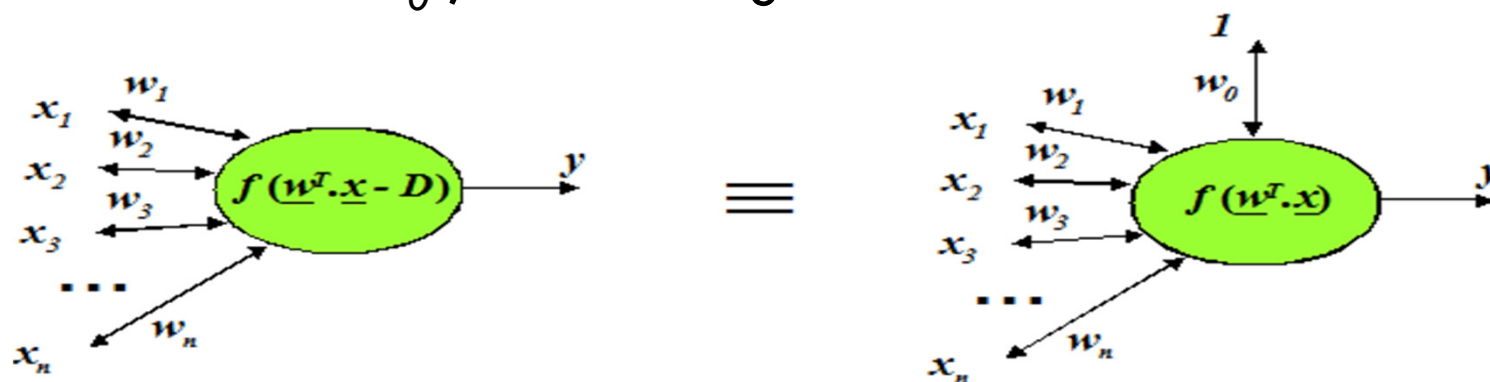
15

- ❑ Alguns componentes merecem atenção neste modelo:
 - ❑ A função $f(\cdot)$ é comumente chamada de Função de Ativação do neurônio;
 - ❑ O argumento da função de ativação, denotado por “*net*”, é definida como o produto escalar entre o vetor de pesos e o vetor de entrada. Ele é o análogo matemático do Potencial de Membrana que existe nos neurônios biológicos e que determina o seu grau de ativação, a partir da excitação externa existente;

O Neurônio Artificial: Modelos Matemáticos de Neurônios

16

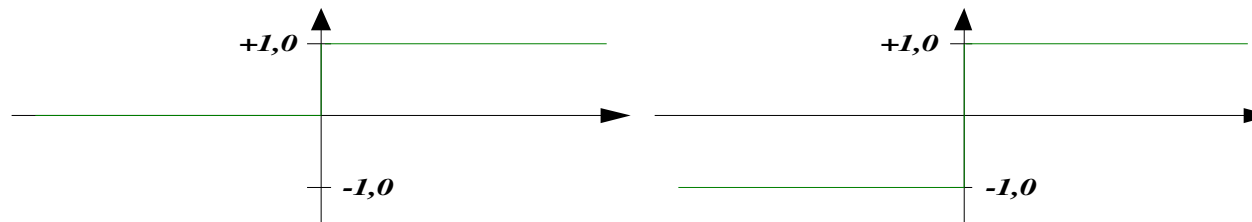
- ❑ O Limiar de Ativação D do neurônio pode ser “embutido” na variável *net* sem perda de generalização, através da sua substituição por uma entrada fixa unitária e seu respectivo peso de conexão w_0 , como na figura abaixo:



O Neurônio Artificial: Modelos Matemáticos de Neurônios

17

- ❑ Muitas funções de ativação diferentes têm sido testadas ao longo dos anos, mas apenas algumas encontraram aplicações práticas relevantes. Entre estas, pode-se citar:
- ❑ **As funções de limiar estrito, como o degrau e o sinal(.) .**



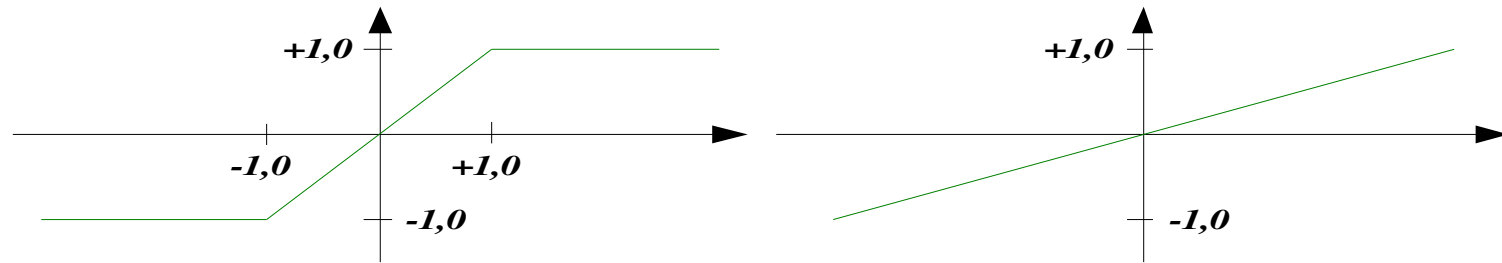
$$y = \begin{cases} 1, & \text{se } \mathbf{net} \geq 0 \\ 0, & \text{se } \mathbf{net} < 0 \end{cases}$$

$$y = \begin{cases} 1, & \text{se } \mathbf{net} \geq 0 \\ -1, & \text{se } \mathbf{net} < 0 \end{cases}$$

O Neurônio Artificial: Modelos Matemáticos de Neurônios

18

- As funções lineares com e sem saturação da saída.



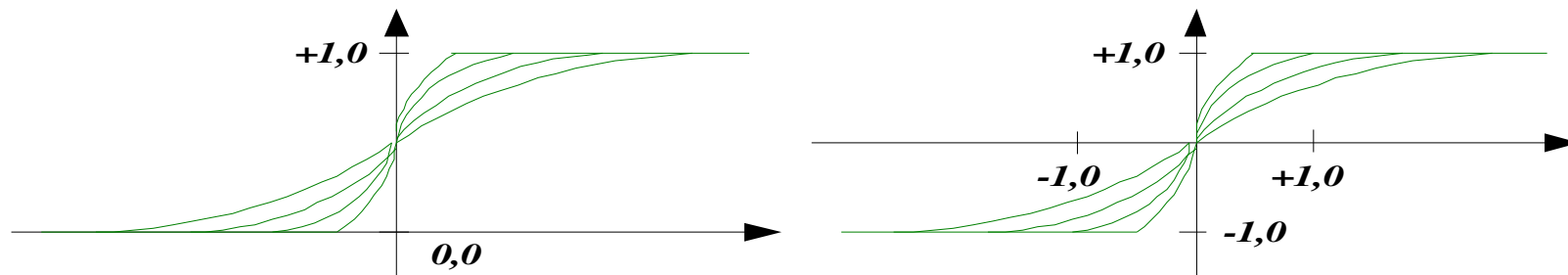
$$y = \begin{cases} 1, & \text{se } net > 1 \\ net, & \text{se } -1 < net < 1 \\ -1, & \text{se } net < -1 \end{cases}$$

$$y = \lambda.net$$

O Neurônio Artificial: Modelos Matemáticos de Neurônios

19

□ *As funções sigmoidais unipolares e bipolares.*



$$y = \frac{1}{1 + e^{-\lambda \cdot net}} \quad y = \frac{2}{1 + e^{-\lambda \cdot net}} - 1$$

Modelos de Redes Neurais Artificiais:

Arquiteturas

20

- ❑ Uma RNA pode ser definida como uma topologia interconectada de neurônios artificiais, na qual, tipicamente, se pode identificar neurônios de entrada, neurônios internos e neurônios de saída.
- ❑ A maneira na qual estes neurônios estão organizados e conectados depende da arquitetura da rede. Em geral, existem três classes fundamentais de arquiteturas de RNA:

Modelos de Redes Neurais Artificiais:

Arquiteturas

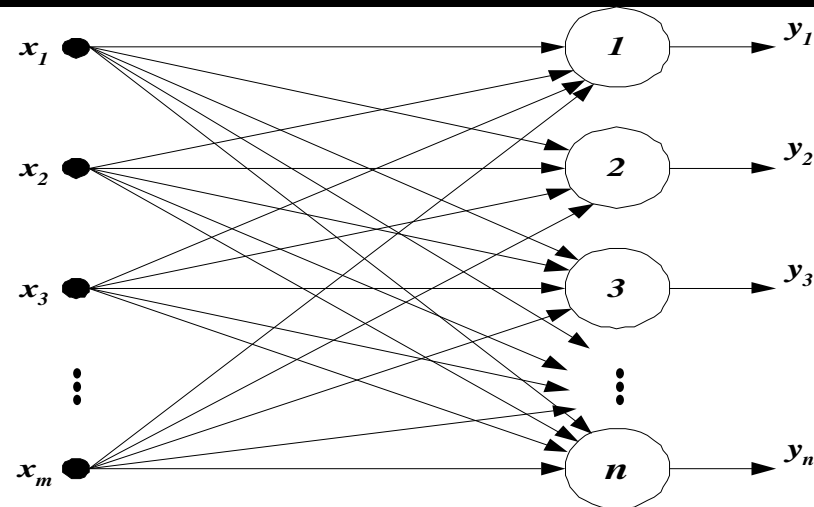
21

- ❑ Redes de Camada Simples:
 - ❑ É a forma mais simples de organização de neurônios em camadas. Existe uma camada de entrada composta por nós de distribuição que se conectam com uma camada de saída composta de neurônios (ou nós de processamento).
 - ❑ O fluxo de sinal através da rede é unidirecional, dos nós de distribuição para os nós de processamento, em direção às saídas da rede. A figura a seguir ilustra esta arquitetura.

Modelos de Redes Neurais Artificiais:

Arquiteturas

22



- ❑ Os nós de distribuição não são considerados como uma camada, pois não ocorre neles processamento dos dados de entrada.

Modelos de Redes Neurais Artificiais:

Arquiteturas

23

- Considere uma rede de camada simples composta por n neurônios recebendo sinais de m entradas, como no diagrama anterior. Os vetores de entrada e de saída são:

$$\underline{x} = [x_1 \quad x_2 \quad x_3 \quad \dots \quad x_m]^T \quad \text{e} \quad \underline{y} = [y_1 \quad y_2 \quad y_3 \quad \dots \quad y_n]^T.$$

Modelos de Redes Neurais Artificiais:

Arquiteturas

24

- ❑ Por sua vez, cada peso w_{ij} conecta o i -ésimo neurônio na entrada j . Então, a notação de pesos com índice duplo significa que a primeira letra do índice identifica o nó de destino e a segunda letra identifica o nó de origem de cada peso.

Modelos de Redes Neurais Artificiais:

Arquiteturas

25

- Assim, a regra de ativação para o i -ésimo neurônio desta rede será:

$$net_i = \sum_{j=1}^m w_{ij} \cdot x_j \quad \text{para } i = 1, 2, 3, \dots, n.$$

Modelos de Redes Neurais Artificiais:

Arquiteturas

26

- Finalmente, a saída de cada neurônio da rede pode ser obtida através de:

$$y_i = f(w_i^T x) \quad \text{para} \quad i = 1, 2, 3, \dots, n$$

$$\text{onde} \quad \underline{w}_i = [w_{i1} \quad w_{i2} \quad w_{i3} \quad \dots \quad w_{im}]^T$$

Modelos de Redes Neurais Artificiais:

Arquiteturas

27

- ❑ Introduzindo o operador matricial Γ , o mapeamento do espaço de entrada \underline{x} no espaço de saída \underline{y} , implementado pela rede, pode ser expresso por:

$$\underline{y} = \Gamma[\underline{W} \cdot \underline{x}].$$

Modelos de Redes Neurais Artificiais:

Arquiteturas

28

- Na equação acima, W chamada de Matriz de Pesos, ou Matriz de Conexões, e o operador $\Gamma [.]$ são expressos por:

$$\underline{W} = \begin{bmatrix} w_{11} & w_{12} & w_{13} & \cdots & w_{1m} \\ w_{21} & w_{22} & w_{23} & \cdots & w_{2m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ w_{n1} & w_{n2} & w_{n3} & \cdots & w_{nm} \end{bmatrix}$$

$$\Gamma [.] = \begin{bmatrix} f(.) & 0 & 0 & \cdots & 0 \\ 0 & f(.) & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & f(.) \end{bmatrix}$$

Modelos de Redes Neurais Artificiais:

Arquiteturas

29

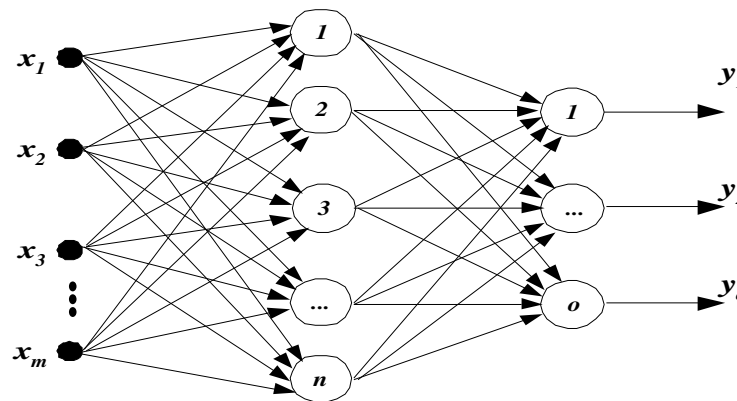
- ❑ Redes de Múltiplas Camadas:
 - ❑ Se distinguem das anteriores pela presença de uma ou mais camadas internas (*hidden layers*), nas quais os neurônios de processamento são chamados comumente de neurônios invisíveis (*hidden neurons*).

Modelos de Redes Neurais Artificiais:

Arquiteturas

30

- ❑ Possui fluxo unidirecional de sinal, a exemplo das redes de camada simples. O diagrama abaixo ilustra esta arquitetura:



Modelos de Redes Neurais Artificiais:

Arquiteturas

31

- ❑ Considerando cada camada em separado, com a saída de uma camada servindo como entrada para a próxima camada, o processamento de sinais em uma rede com k camadas pode ser expresso por:

$$\underline{h}_1 = \Gamma_1 [\underline{W}_1 \cdot \underline{x}],$$

$$\underline{h}_2 = \Gamma_2 [\underline{W}_2 \cdot \underline{h}_1],$$

, ...,

$$\underline{y} = \Gamma_k [\underline{W}_k \cdot \underline{h}_{k-1}].$$

Modelos de Redes Neurais Artificiais:

Arquiteturas

32

❑ Redes Recorrentes:

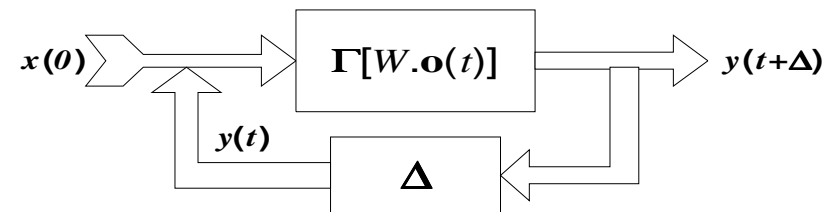
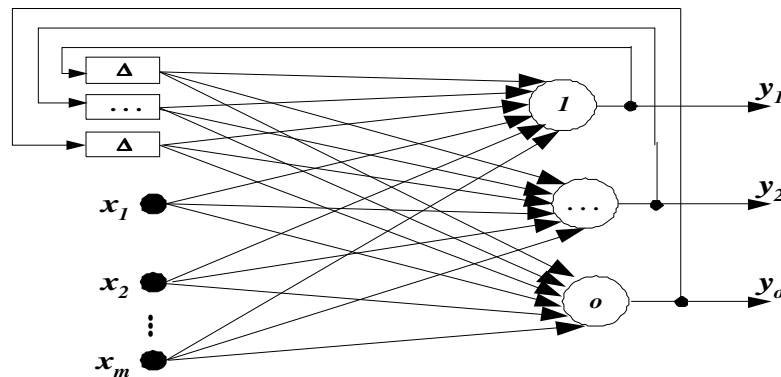
- ▣ Uma rede recorrente pode ser obtida a partir de uma rede convencional de uma ou mais camadas através da criação de uma rota de realimentação conectando os neurônios de saída aos neurônios de entrada.

Modelos de Redes Neurais Artificiais:

Arquiteturas

33

- ❑ A rede resultante é mostrada abaixo, através de um diagrama de interconexões e de um diagrama em blocos:



Modelos de Redes Neurais Artificiais:

Arquiteturas

34

- ❑ O tempo de atraso entre os instantes t e $t+\Delta$ é introduzido pelo operador de atraso Δ . Este atraso tem um significado simbólico, sendo uma analogia ao período de propagação de sinal existente em um neurônio biológico. Em aplicações práticas, ele pode refletir o intervalo de amostragem, ou seja, o intervalo de tempo entre duas medições simultâneas de um sinal.

Modelos de Redes Neurais Artificiais:

Arquiteturas

35

- ❑ Usando a mesma notação empregada anteriormente, pode-se expressar o mapeamento que ocorre em uma rede recorrente por:

$$\underline{y}^{(t+\Delta)} = \Gamma[\underline{W} \cdot \underline{y}^{(t)}].$$

- ❑ Note que a entrada \underline{x} só é necessária para a inicialização da rede, de modo que $\underline{y}(0) = \underline{x}(0)$. Depois deste momento, a rede processa os sinais de forma autônoma, até atingir a estabilidade.

Processamento Neural

36

- ❑ A discussão sobre os modelos de RNA até agora estava focada principalmente no cálculo das saídas de cada arquitetura para um dado vetor de entrada x . Este procedimento de cálculo ou de processamento dos sinais em uma rede é comumente denotado pelo termo em língua inglesa “*Recall*”, que significa “Lembrança, Recordação”.

Processamento Neural

37

- ❑ Em outras palavras, o processamento de uma RNA para um dado vetor de entrada e um conjunto de matrizes de pesos pode ser visto como uma recordação do que ela “aprendeu” em sua fase de treinamento, quando estes pesos foram ajustados. É a decodificação de um conteúdo (ou conhecimento) que foi armazenado e codificado na forma de um conjunto de pesos para as conexões desta rede.

Processamento Neural

38

- A partir de agora, será analisado de forma funcional o “*recall*” de uma RNA, e as tarefas que ele pode executar. As várias tarefas podem ser vistas como diferentes formas de Processamento Neural de Informações .

Processamento Neural

39

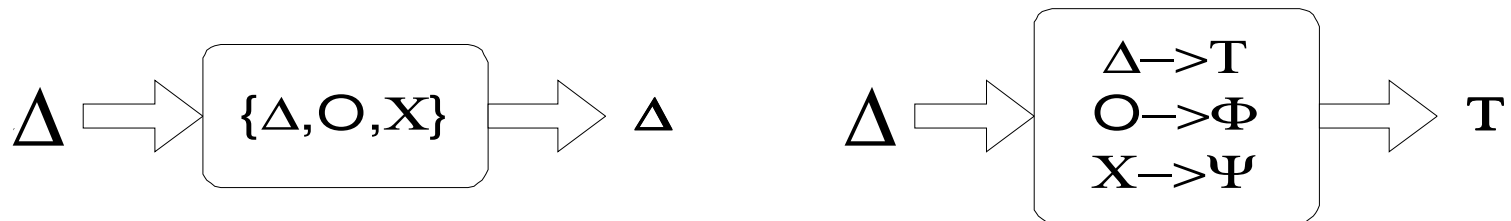
❑ Memórias Associativas:

- ❑ Assuma que um certo número de padrões possa ser de alguma forma armazenado no interior de uma RNA. Depois, se for apresentado na entrada da rede um padrão similar a um dos padrões armazenados, esta rede pode associar este padrão com o padrão similar que ela “conhece”. Este processo é chamado de Auto-Associação .

Processamento Neural

40

- Por outro lado, padrões de entrada também podem ser armazenados de forma hétero-associativa, ou seja, são armazenadas associações entre pares de padrões.



Processamento Neural

41

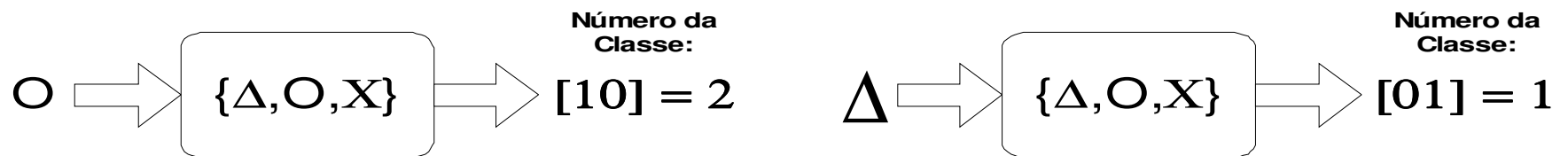
❑ Classificação:

- ❑ Assuma agora que um conjunto de padrões de entrada é dividido em um número de classes ou categorias, que são então armazenadas de algum modo em uma RNA.
- ❑ Então, um processamento neural de classificação responde a um padrão de entrada pela definição ou recordação da pertinência deste padrão a uma das categorias “aprendidas”.

Processamento Neural

42

- Tipicamente, estas classes são codificadas através de vetores de dados, de modo que uma rede de classificação normalmente emprega saídas binárias. Este tipo de processamento neural é ilustrado a seguir.



Processamento Neural

43

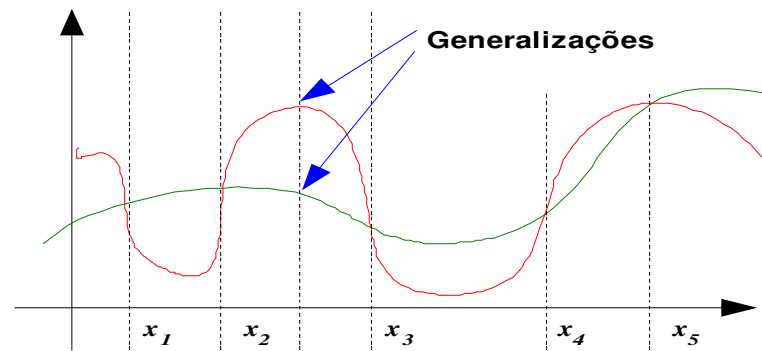
❑ Aproximação de Funções:

Duas propriedades muito importantes de RNA, que as fazem particularmente importantes para o emprego em aplicações complexas são:

- ❑ sua capacidade de aproximar funções e mapeamentos arbitrários de dados;
- ❑ Sua capacidade de generalização, ou seja, de interpolação de padrões de saída a partir de padrões de entrada desconhecidos.

Processamento Neural

44



Aprendizado e Adaptação em RNA

45

❑ ○ Perceptron

- ❑ Em 1958, foi proposto por Franklin Rosenblatt o primeiro modelo neural com capacidade de aprendizado, chamado de *Perceptron*. Nas palavras de Rosenblatt, ele consistia de “uma máquina adaptativa capaz de classificar padrões de entrada pela modificação das conexões de entrada nos neurônios com funções de limiar na saída”.

Aprendizado e Adaptação em RNA

46

- Em outras palavras, o modelo era baseado no neurônio MCP, com um combinador linear de entrada seguido de uma função de ativação de limiar estrito na saída. Sua função era a classificação dos padrões de entrada codificados nas entradas em uma de duas classes possíveis, C_1 e C_2 .

Aprendizado e Adaptação em RNA

47

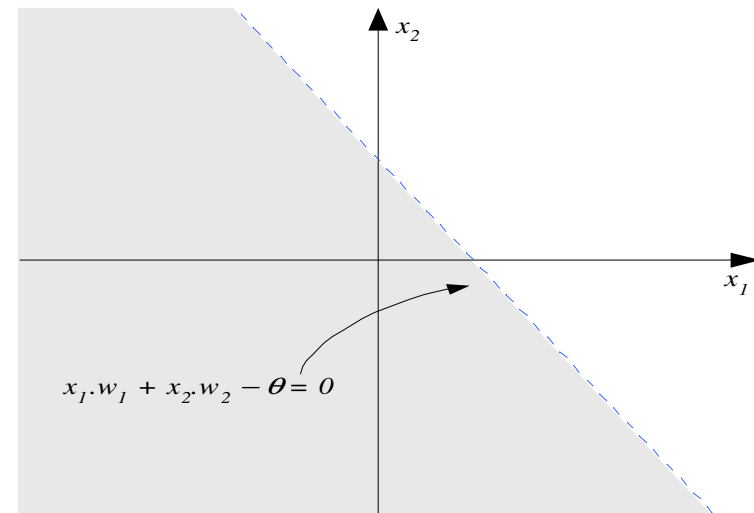
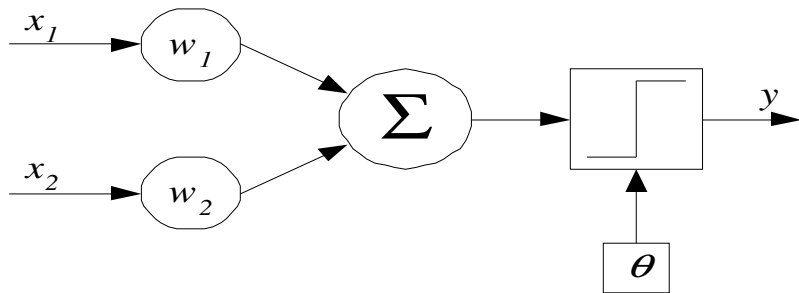
- Assim, no caso de um perceptron elementar, ou seja, de uma rede formada por um único neurônio, o espaço de entrada n - dimensional é dividido por um hiperplano em duas regiões de decisão. Este hiperplano é definido pela função

$$\sum_{i=1}^n x_i \cdot w_i - \theta = 0$$

- Para o caso particular de duas entradas, x_1 e x_2 , a fronteira de decisão tem a forma de uma linha reta como mostrado a seguir.

Aprendizado e Adaptação em RNA

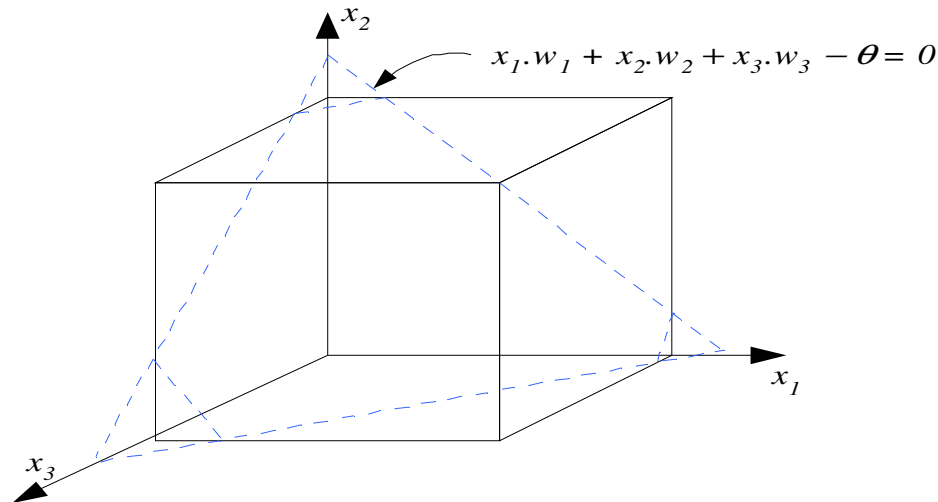
48



Aprendizado e Adaptação em RNA

49

- ❑ Para o caso particular de três entradas, x_1 , x_2 e x_3 , a fronteira de decisão tem a forma de um plano de separação, como mostrado abaixo.



Aprendizado e Adaptação em RNA

50

❑ Aprendizado no Perceptron:

- ❑ O processo de escolha da superfície de separação determinada por um perceptron elementar é comumente chamado de APRENDIZADO, ou TREINAMENTO, ou ainda ADAPTAÇÃO do modelo às classes que se quer separar.

Aprendizado e Adaptação em RNA

51

- ❑ Esta escolha ocorre por ajustes sucessivos nos pesos de conexão das entradas do Perceptron. Estes ajustes são calculados para reduzir a diferença entre a saída desejada e a saída real do neurônio. Este processo é particularmente simples no caso de um Perceptron elementar.
- ❑ A existência de um conjunto de dados de entrada e sua respectiva classificação (correspondendo à saída desejada do Perceptron) é necessária para que ocorra este ajuste. Este tipo de aprendizado, baseado na aproximação entre um comportamento ideal conhecido e um comportamento real ajustável, é denominado APRENDIZADO SUPERVISIONADO.

Aprendizado e Adaptação em RNA

52

- Considere que, em uma iteração p de aprendizado, a saída desejada para um Perceptron é $y_D(p)$ e a saída real é $y(p)$. Então, o erro é dado por

$$e(p) = y_D(p) - y(p).$$

- Iteração p aqui significa que o p -ésimo dado de treinamento está sendo apresentado ao Perceptron.

Aprendizado e Adaptação em RNA

53

- ❑ Se o erro $e(p)$ é positivo, significa que é necessário aumentar a saída $y(p)$ do Perceptron. Por outro lado, se o erro é negativo, então é necessário diminuir a saída $y(p)$. Considerando que cada entrada x_i contribui para a saída com uma parcela igual a $x_i(p) \cdot w_i(p)$, observa-se que:
 - ❑ Se esta entrada é positiva, um aumento no valor de w_i vai significar um aumento na saída do Perceptron; e
 - ❑ Se esta entrada é negativa, um aumento no valor de w_i vai significar uma diminuição na saída do Perceptron.

Aprendizado e Adaptação em RNA

54

- Assim, a Regra de Aprendizado do Perceptron ou Regra Delta é definida por:

$$w_i(p + 1) = w_i(p) + \alpha \cdot x_i(p) \cdot e(p).$$

- onde α é a Taxa de Aprendizado, um valor constante e positivo menor que a unidade, que determina a velocidade de ajuste dos pesos da rede.
- Esta regra, a primeira tentativa de inserção de aprendizado em um sistema computacional, foi proposta em 1958 por Rosenblatt (e em outro contexto por Bernard Widrow em 1960) e serviu como base para o desenvolvimento de outros tipos de aprendizado.

Aprendizado e Adaptação em RNA

55

- ❑ A seguir, é mostrado um algoritmo de treinamento de um Perceptron, empregado para tarefas de classificação:
 - ❑ **Passo 1 – Inicialização:**
 - Iniciar de forma aleatória os pesos w_i e a variável de limiar θ na faixa $[-0,5 ; 0,5]$;
 - ❑ **Passo 2 – Ativação:**
 - Processamento do Perceptron pela aplicação das entradas x_i e cálculo da saída y , iteração 1 ($p=1$):

$$y(p) = \text{sinál} \left(\sum_{i=1}^n x_i(p) \cdot w_i(p) - \theta \right)$$

Aprendizado e Adaptação em RNA

56

□ **Passo 3 – Ajuste dos Pesos:**

- Cálculo do erro cometido para a iteração atual:

$$e(p) = y_D(p) - y(p).$$

- Atualização dos pesos do Perceptron:

$$w_i(p + 1) = w_i(p) + \alpha \cdot x_i(p) \cdot e(p).$$

Aprendizado e Adaptação em RNA

57

❑ ***Passo 4 – Repetição Iterativa:***

- Incremento unitário no contador de iteração p ;
- Retorno ao passo 2;
- Repetição do processo até que a convergência dos valores dos pesos seja alcançada.
- Exemplo: O Perceptron executando funções Booleanas .
- As tabelas-verdade das funções E, OU, e X-OU (ou exclusivo) são mostradas abaixo.

Aprendizado e Adaptação em RNA

58

Variáveis de Entrada		E	OU	X-OU
x_1	x_2	$x_1 \cap x_2$	$x_1 \cup x_2$	$x_1 \oplus x_2$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Aprendizado e Adaptação em RNA

59

- Aplicando o algoritmo anterior e usando os valores da tabela-verdade da função “E” como dados de treinamento, os pesos de um Perceptron Elementar são ajustados e convergem para um conjunto de valores adequado, como mostra a tabela abaixo (OBS: $\theta = 0,2$ e $\alpha = 0,1$):

Aprendizado e Adaptação em RNA

60

Época	Entradas		Y desej.	Pesos Início		Y real	Erro	Pesos Fim	
	x_1	x_2	y_D	w_1	w_2	y'	e	w_1	w_2
1	0	0	0	0,3	-0,1	0	0	0,3	-0,1
	0	1	0	0,3	-0,1	0	0	0,3	-0,1
	1	0	0	0,3	-0,1	1	-1	0,2	-0,1
	1	1	1	0,2	-0,1	0	1	0,3	0,0
2	0	0	0	0,3	0,0	0	0	0,3	0,0
	0	1	0	0,3	0,0	0	0	0,3	0,0
	1	0	0	0,3	0,0	1	-1	0,2	0,0
	1	1	1	0,2	0,0	1	0	0,2	0,0
3	0	0	0	0,2	0,0	0	0	0,2	0,0
	0	1	0	0,2	0,0	0	0	0,2	0,0
	1	0	0	0,2	0,0	1	-1	0,1	0,0
	1	1	1	0,1	0,0	0	1	0,2	0,1
4	0	0	0	0,2	0,1	0	0	0,2	0,1
	0	1	0	0,2	0,1	0	0	0,2	0,1
	1	0	0	0,2	0,1	1	-1	0,1	0,1
	1	1	1	0,1	0,1	1	0	0,1	0,1
5	0	0	0	0,1	0,1	0	0	0,1	0,1
	0	1	0	0,1	0,1	0	0	0,1	0,1
	1	0	0	0,1	0,1	0	0	0,1	0,1
	1	1	1	0,1	0,1	1	0	0,1	0,1

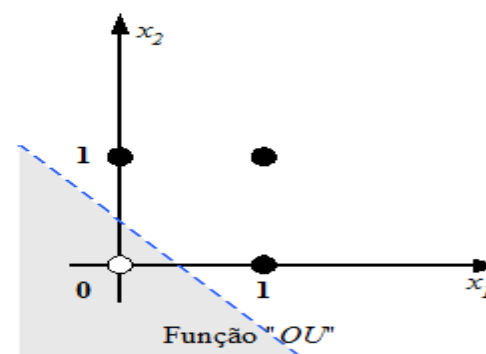
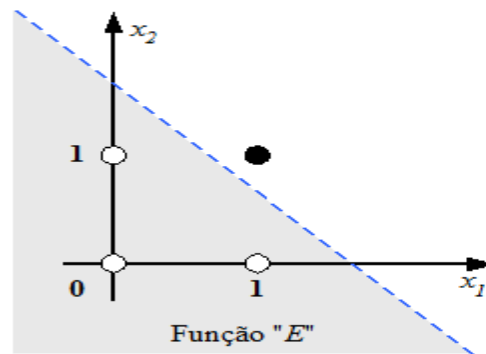
Aprendizado e Adaptação em RNA

61

- ❑ De maneira similar, é possível treinar um Perceptron Elementar para que ele “aprenda” a executar a função “OU”.
- ❑ No entanto, um Perceptron Elementar não pode ser treinado para executar a função “X-OU”. Isto ocorre porque a função “X-OU” não é LINEARMENTE SEPARÁVEL, como são as funções “E” e “OU”.
- ❑ Em outras palavras, é impossível traçar uma linha reta que separe as regiões em que a função “X-OU” possui valor unitário das regiões em que a função assume valor nulo.
- ❑ Os gráficos abaixo ilustram esta limitação:

Aprendizado e Adaptação em RNA

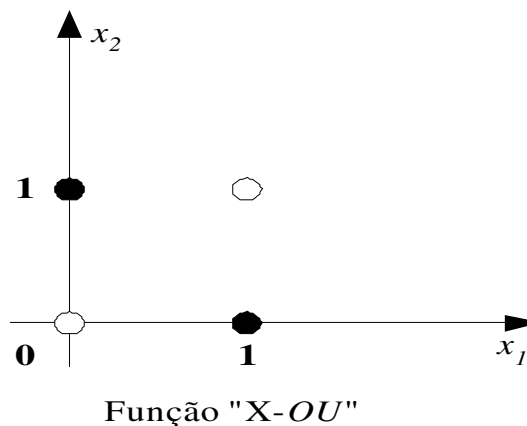
62



Aprendizado e Adaptação em RNA

63

- ❑ Um Perceptron Elementar é capaz de representar uma função apenas se ela for linearmente separável (LS).
- ❑ Pode-se mostrar que RNA de uma única camada são capazes de representar apenas funções LS, mesmo se a função de ativação for substituída por uma função mais complexa (p. ex. sigmoidal).



RNA Não Lineares

64

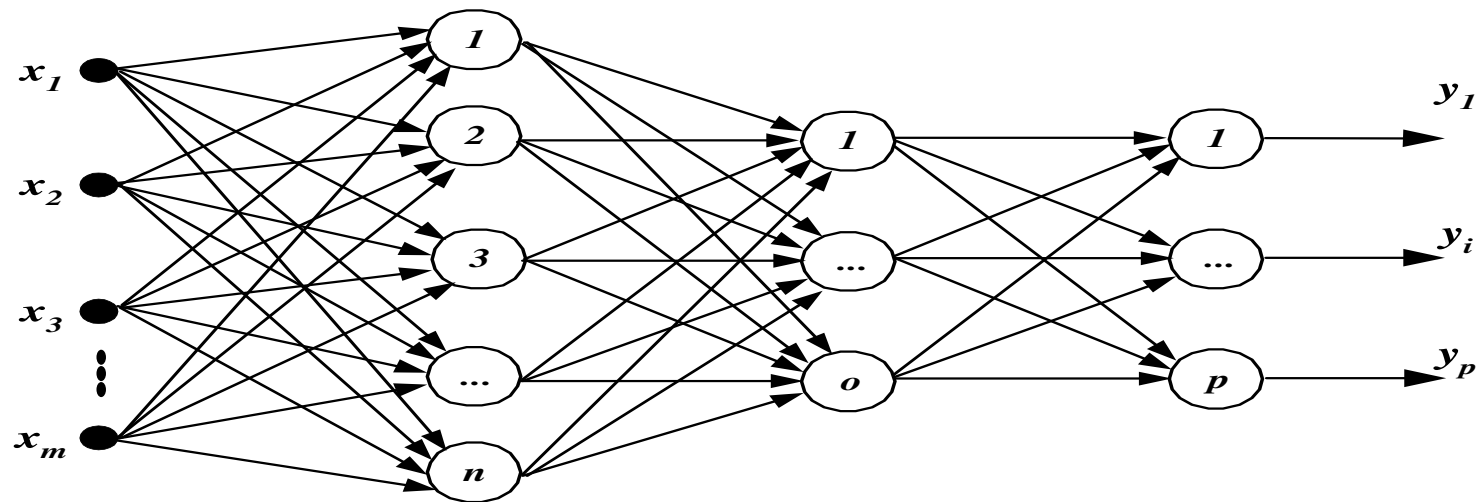
❑ Redes de Múltiplas Camadas:

▣ Uma RNA de múltiplas camadas, comumente chamada de rede MLP (Multi-Layer Perceptron) possui:

- fluxo de sinal unidirecional;
- uma camada de entrada;
- uma ou mais camadas escondidas; e
- uma camada de saída.

RNA Não Lineares

65



RNA Não Lineares

66

- A vantagem principal das redes MLP é a sua capacidade de aproximação de funções não lineares. Em 1988, George Cybenko mostrou que uma rede MLP com uma camada escondida consegue aproximar qualquer função contínua em um número finito de iterações de treinamento. De modo similar, ele mostrou que uma rede MLP com duas camadas escondidas é capaz de aproximar qualquer função não linear, mesmo que descontínua.

RNA Não Lineares

67

- ▣ Vários trabalhos demonstraram no fim da década de 80 a capacidade de aproximação universal das redes MLP (Hornik 1989, Girosi & Poggio 1990).

Aprendizado e Adaptação

68

- ❑ Um algoritmo de treinamento de uma RNA é sempre determinado:
 - ▣ Pela conexão dos neurônios, ou topologia da rede;
 - ▣ Pela função de ativação empregada em cada neurônio;
 - ▣ Pela Lei de Aprendizado escolhida.

Aprendizado e Adaptação

69

- ❑ O Aprendizado em redes MLP ocorre de modo similar ao aprendizado no Perceptron:
 - ▣ Um conjunto de padrões de treinamento, composto de vetores de entrada e as respectivas saídas desejadas da rede para estes vetores, são apresentados à rede MLP;
 - ▣ É calculada a saída da rede e o erro entre a saída desejada e a saída obtida é usado como ajuste para o aprendizado.

Aprendizado e Adaptação

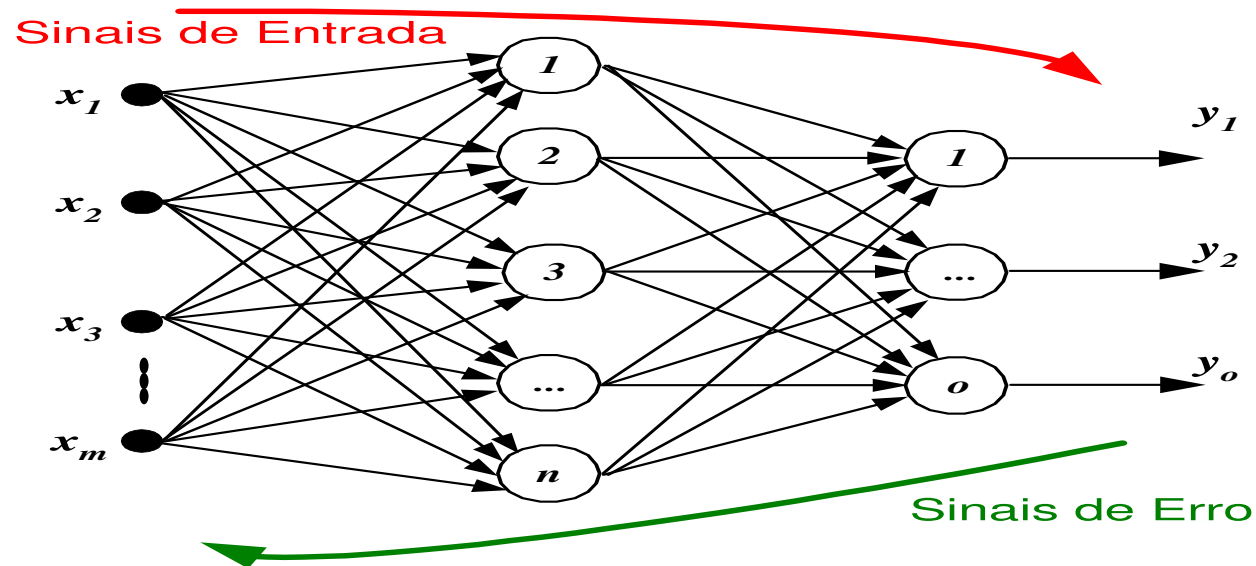
70

- Mas, no Perceptron, existia apenas uma conexão a ser ajustada para cada entrada, e um único neurônio de saída. Agora, existem muitas conexões em várias camadas, e cada conexão pode contribuir para mais de uma saída da rede. Além disso, é comum o emprego de funções de ativação não-lineares nos neurônios de uma rede MLP, o que dificulta um pouco mais a obtenção do algoritmo de treinamento adequado.
- Tipicamente, os neurônios de uma dada camada em uma rede MLP são completamente conectados, ou seja, cada neurônio desta camada é conectado a cada um dos neurônios da camada subsequente.
- Não existe conexão entre neurônios de camadas não adjacentes, e nem conexões entre neurônios em uma mesma camada.

Aprendizado e Adaptação

71

- Para a derivação do algoritmo de treinamento “*Back-Propagation*”, será considerada uma rede MLP de 3 camadas. Os índices i , j e k se referem a neurônios nas camadas de entrada, escondida e de saída, respectivamente.



Aprendizado e Adaptação

72

- ▣ Os sinais de entrada, x_1, x_2, \dots, x_m são propagados através da rede da esquerda para a direita, enquanto os sinais de erro, e_1, e_2, \dots, e_o são propagados da direita para a esquerda, ou seja, da camada de saída para a camada de entrada.
- ▣ A propagação dos sinais de erro inicia na camada de saída em direção da camada escondida. Na iteração de treinamento p , o sinal de erro na saída do neurônio k é definida por:

$$e_k(p) = y_{Dk}(p) - y_k(p),$$

$$w_{kj}(p+1) = w_{kj}(p) + \Delta w_{kj}(p).$$

onde $y_{Dk}(p)$ é a saída desejada para o neurônio k na iteração p .

Aprendizado e Adaptação

73

- No caso da camada de saída, a correção em cada peso é similar à da regra Delta, aplicada no Perceptron:
 - No caso do Perceptron, era usada a entrada do neurônio, x_i , na correção do peso. Mas agora, na rede MLP, as entradas dos neurônios da camada de saída são diferentes das entradas da rede na camada inicial.
 - Então, é empregada a saída do neurônio j , da camada escondida, ao invés da entrada x_i :

$$\Delta w_{kj}(p) = \alpha \cdot y_j(p) \cdot \delta_k(p),$$

Aprendizado e Adaptação

74

- ▣ onde $\delta_k(p)$ é o gradiente do erro cometido no neurônio k da camada de saída na iteração de treinamento p .
- ▣ O gradiente do erro em um dado neurônio da camada de saída é definido como a derivada da função de ativação deste neurônio multiplicada pelo erro em sua saída:

$$\delta_k(p) = f'(X_k(p)) \cdot e(p).$$

Aprendizado e Adaptação

75

- Acima, $X_k(p)$ é a soma ponderada de entrada do neurônio k , na iteração p :

$$X_k(p) = \sum_{j=0}^n x_{jk}(p) \cdot w_{jk}(p).$$

- Considerando uma função de ativação sigmoideal unipolar, tem-se:

$$y_k(p) = \frac{1}{1 + e^{-X_k(p)}} \quad \Rightarrow \quad y'_k(p) = y_k(p) \cdot (1 - y_k(p)),$$

e o gradiente se torna:

$$\delta_k(p) = y_k(p) \cdot (1 - y_k(p)) \cdot e(p).$$

Aprendizado e Adaptação

76

- ▣ Para o cálculo das correções nos pesos da camada oculta, é empregada a mesma equação da camada de saída:

$$\Delta w_{ji}(p) = \alpha \cdot x_i(p) \cdot \delta_j(p),$$

onde $\delta_j(p)$ é o gradiente do erro cometido no neurônio j da camada oculta na iteração de treinamento p .

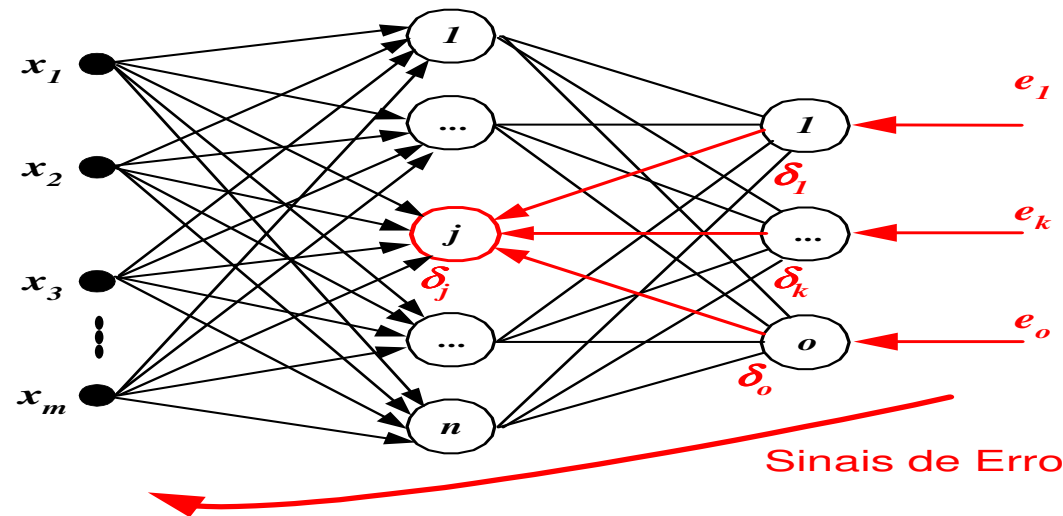
- ▣ Considerando os neurônios da camada oculta também com funções de ativação sigmoidais unipolares, este gradiente do erro, correspondendo a um dado neurônio j da camada oculta, é

$$\delta_j(p) = y_j(p) \cdot (1 - y_j(p)) \cdot \sum_{k=0}^o \delta_k(p) \cdot w_{kj}(p).$$

Aprendizado e Adaptação

77

- Note que, para o cálculo do gradiente do erro para os neurônios de uma dada camada, é usada uma soma dos gradientes de erro da camada posterior, ponderada pelos pesos de conexão entre estas duas camadas.



Aprendizado e Adaptação

78

- ▣ Esta equação reflete muito bem a denominação do algoritmo de treinamento, “*Back-Propagation*”, já que o erro está sendo propagado da camada de saída para trás, em direção da camada de entrada.
- ▣ A seguir, é mostrado um algoritmo de treinamento usando o método de *Back-Propagation*:
 - Passo 1 – Inicialização:
 - Iniciar de forma aleatória os pesos w_{ij} e as variáveis de limiar θ_i uniformemente distribuídos em uma faixa pré-definida de valores $[-2,4 / (m+n+o); 2,4 / (m+n+o)]$;

Aprendizado e Adaptação

79

■ Passo 2 – Ativação:

- Processamento da rede MLP pela aplicação das entradas x_i e cálculo das saídas y_i , iteração 1 ($p = 1$):
 - Cálculo das saídas da camada escondida. Para o neurônio j desta camada, tem-se:

$$y_j(1) = \frac{1}{1 + e^{-X_i(1)}} \quad \text{onde} \quad X_i(1) = \sum_{j=0}^m x_j(1) \cdot w_{ji}(1),$$

- e m é o número de entradas da rede.

Aprendizado e Adaptação

80

- Cálculo das saídas da rede MLP para todos os neurônios da camada de saída:

$$y_k(1) = \frac{1}{1 + e^{-X_j(1)}} \quad \text{onde} \quad X_j(1) = \sum_{k=0}^n y_j(1) \cdot w_{kj}(1),$$

- e n é o número de neurônios na camada oculta da rede.

Aprendizado e Adaptação

81

■ Passo 3 – Ajuste dos Pesos da Camada de Saída:

- Cálculo do gradiente do erro cometido na iteração atual para os neurônios da camada de saída:

$$\delta_k(p) = y_k(p) \cdot (1 - y_k(p)) \cdot e(p).$$

- Atualização dos pesos da camada de saída:

$$\Delta w_{kj}(p) = \alpha \cdot y_j(p) \cdot \delta_k(p).$$

Aprendizado e Adaptação

82

■ Passo 4 – Ajuste dos Pesos da Camada Oculta:

- Cálculo do gradiente do erro cometido na iteração atual para os neurônios da camada oculta:

$$\delta_j(p) = y_j(p) \cdot (1 - y_j(p)) \cdot \sum_{k=0}^o \delta_k(p) \cdot w_{kj}(p).$$

- Atualização dos pesos da camada oculta:

$$\Delta w_{ji}(p) = \alpha \cdot x_i(p) \cdot \delta_j(p).$$

Aprendizado e Adaptação

83

■ Passo 5 – Repetição Iterativa:

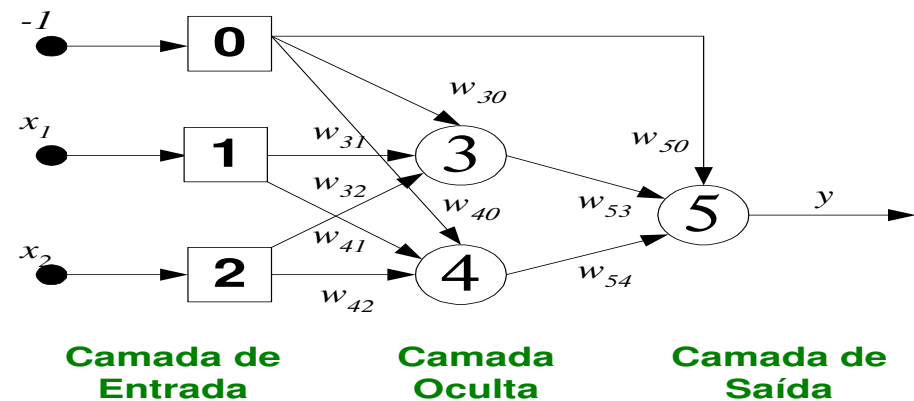
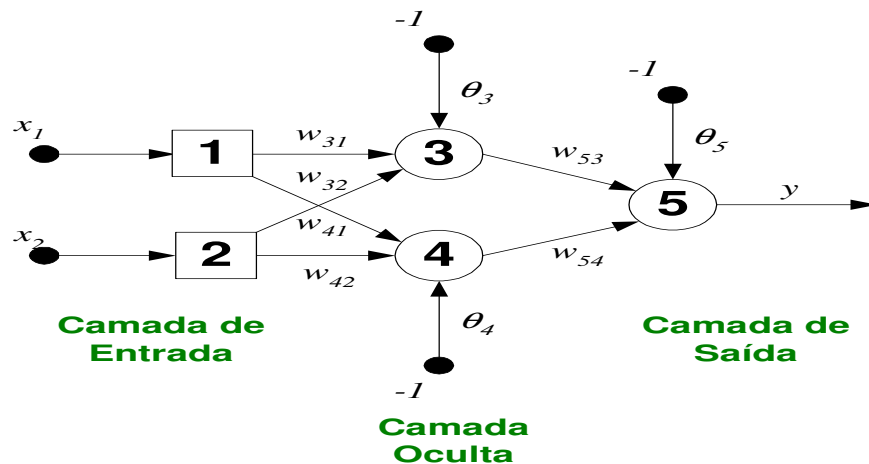
- Incremento unitário no contador de iteração p ;
- Retorno ao passo 2;
- Repetição do processo até que um critério de parada seja alcançado, como:
 - a convergência dos valores dos pesos seja alcançada;
 - Um nível de erro (ou de desempenho) pré-determinado seja alcançado.

Aprendizado e Adaptação

84

❑ Exemplo 1: uma rede MLP para a implementação da Função X-OU

❑ Considere a rede de três camadas mostrada a seguir:



Aprendizado e Adaptação

- ❑ Aplicando o algoritmo de BP descrito anteriormente, um coeficiente de aprendizado $\alpha = 0,1$ e os pesos iniciados aleatoriamente como na linha 0 da tabela abaixo, a sequência de ajustes de pesos para as duas primeiras épocas de treinamento

[illegible]

RNA Não Lineares: Redes de Processamento

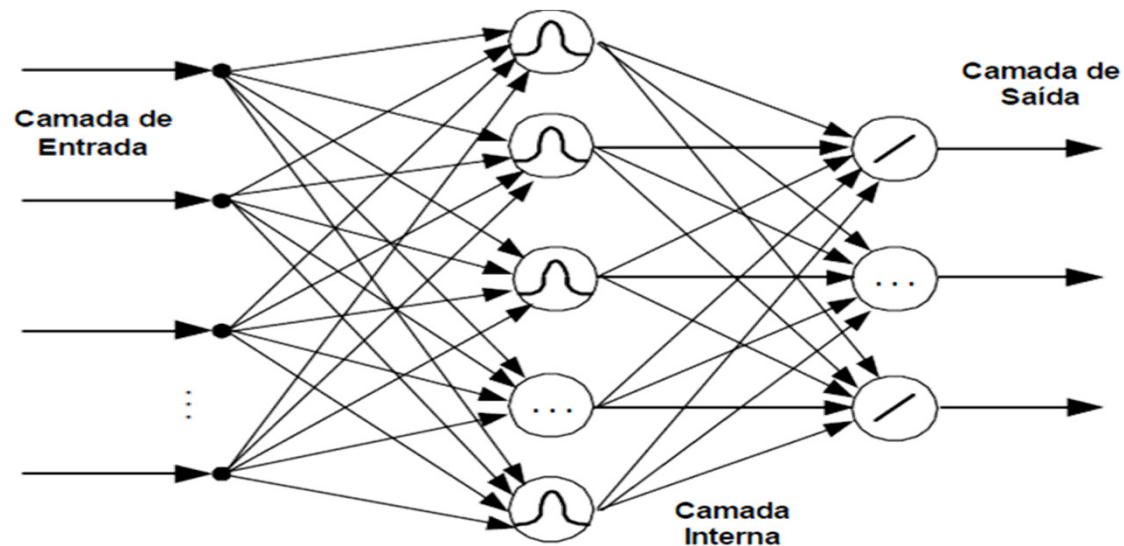
86

- ❑ Nos Perceptrons lineares e não-lineares estudados até aqui, o fluxo de sinais é GLOBAL, ou seja, para qualquer vetor de entrada x aplicado, todos os neurônios da rede vão contribuir para o cálculo do vetor de saída y .
- ❑ Existem algumas arquiteturas de RNA que possuem um fluxo de dados LOCAL, ou seja, apenas um subconjunto dos neurônios é ativado para um dado vetor de entrada x . Apenas estes neurônios ativos irão contribuir para o cálculo do vetor de saída y .
- ❑ Os neurônios desta classe de redes são comumente chamados de Campos Receptivos, e possuem uma inspiração biológica baseada no funcionamento de algumas regiões do córtex cerebral e do cerebelo, entre outras.

Redes Neurais de Base Radial (RBF)

87

- ❑ É mostrado abaixo um diagrama típico de uma rede neural de base radial.



Redes Neurais de Base Radial (RBF)

88

- ❑ Alguns trabalhos demonstraram, a partir da década de 80, a capacidade de aproximação universal das redes RBF.
- ❑ Os níveis de ativação dos neurônios da camada interna são dados por:

$$g_i = R_i(\underline{x}) = R_i\left(\frac{\|\underline{x} - \underline{u}_i\|}{\sigma_i}\right) \quad i = 1, 2, 3, \dots, N,$$

- onde \underline{x} é o vetor de entrada, \underline{u}_i é um vetor com a mesma dimensão de \underline{x} , N é o número de neurônios ou campos receptivos na camada interna da rede, $R_i(.)$ é a i -ésima função de base radial com máximo simples na origem e σ_i é o comprimento ou fator de escala da função $R_i(.)$.

Redes Neurais de Base Radial (RBF)

89

- ❑ O operador $\| \cdot \|$ comumente denota a distância euclidiana entre dois vetores, e o vetor \underline{u}_i representa o centro da função de base radial $R_i(\cdot)$.

- ❑ Tipicamente, $R_i(\cdot)$ é uma função Gaussiana, representada por:

$$R_i(\underline{x}) = \exp \left(- \frac{\|\underline{x} - \underline{u}_i\|^2}{2\sigma_i^2} \right)$$

- ❑ ou uma função logística como

$$R_i(\underline{x}) = \frac{1}{1 + \exp \left(\frac{\|\underline{x} - \underline{u}_i\|^2}{2\sigma_i^2} \right)}$$

Redes Neurais de Base Radial (RBF)

90

- Assim, o nível máximo de ativação é obtido quando o vetor de entrada \underline{x} está localizado no centro \underline{u}_i da função $R_i(.)$. Finalmente, a saída de uma rede RBF é calculada pela soma ponderada das ativações dos neurônios da camada interna:

$$y(\underline{x}) = \sum_{i=1}^N w_i R_i(\underline{x})$$

Interpolação e Aproximação de Funções com Redes RBF

91

- Assumindo que não há ruído no processo de obtenção de um conjunto de dados de treinamento, deseja-se estimar uma rede RBF que resulta nas saídas exatas para todos os pontos deste conjunto de treinamento.
- Este problema é chamado de **Interpolação RBF**, e a rede empregada deve possuir o mesmo número de campos receptivos que o número de pontos de treinamento, ou seja, cada neurônio da camada interna corresponde a um dado padrão de treinamento existente.
- Considerando funções de base Gaussianas, cada entrada \underline{x}_i servirá como centro da função de base R_i , e portanto a rede RBF resultante será

$$y(\underline{x}) = \sum_{i=1}^N w_i \exp \left(- \frac{\|\underline{x} - \underline{x}_i\|^2}{2\sigma_i^2} \right).$$

Interpolação e Aproximação de Funções com Redes RBF

92

- Dados os comprimentos σ_i , $i=1,..,N$, pode-se obter as seguintes equações simultâneas em função dos pesos de saída, w_i :

$$\begin{cases} y_1 = w_1 \exp \left(- \frac{\| \underline{x}_1 - \underline{x}_1 \|^2}{2 \sigma_1^2} \right) + \dots + w_N \exp \left(- \frac{\| \underline{x}_1 - \underline{x}_N \|^2}{2 \sigma_N^2} \right) \\ \dots \\ y_N = w_1 \exp \left(- \frac{\| \underline{x}_N - \underline{x}_1 \|^2}{2 \sigma_1^2} \right) + \dots + w_N \exp \left(- \frac{\| \underline{x}_N - \underline{x}_N \|^2}{2 \sigma_N^2} \right) \end{cases}$$

Interpolação e Aproximação de Funções com Redes RBF

93

- Reescrevendo de forma compacta, tem-se

$$\begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{bmatrix} = \begin{bmatrix} \exp\left(-\frac{\|\underline{x}_1 - \underline{x}_1\|^2}{2\sigma_1^2}\right) & \dots & \dots & \exp\left(-\frac{\|\underline{x}_1 - \underline{x}_N\|^2}{2\sigma_N^2}\right) \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \exp\left(-\frac{\|\underline{x}_N - \underline{x}_1\|^2}{2\sigma_1^2}\right) & \dots & \dots & \exp\left(-\frac{\|\underline{x}_N - \underline{x}_N\|^2}{2\sigma_N^2}\right) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_N \end{bmatrix}$$

Interpolação e Aproximação de Funções com Redes RBF

94

- ❑ Escrevendo o sistema acima em forma matricial, resulta:

$$\underline{y} = G \underline{w}$$

- ❑ Se a matriz G for não-singular, ela vai admitir inversa e a solução para o sistema acima será única:

$$\underline{w} = G^{-1} \underline{y}$$

- ❑ Quando o número N de pontos no conjunto de treinamento é muito grande, então o número de campos receptivos da rede será menor que N e o problema resultante é chamado de **Aproximação RBF**.

Interpolação e Aproximação de Funções com Redes RBF

95

- ❑ Neste caso, deseja-se encontrar um conjunto de pesos w que minimize a função de custo quadrática. Escrevendo o sistema anterior em forma matricial, resulta:

$$E = \sum_{i=1}^N \left[y_i - \sum_{j=1}^M w_j \exp \left(- \frac{\| \underline{x}_i - \underline{x}_j \|^2}{2 \sigma_j^2} \right) \right]^2.$$

- ❑ onde M é o número de campos receptivos da rede e $M < N$. Considerando a mesma abordagem matricial adotada anteriormente, o problema agora é que a matriz G não é quadrada.

Interpolação e Aproximação de Funções com Redes RBF

96

- Resulta que:

$$\underline{y} = G \underline{w} \Rightarrow G^T \underline{y} = G^T G \underline{w}$$

$$\Rightarrow (G^T G)^{-1} G^T \underline{y} = (G^T G)^{-1} G^T G \underline{w}$$

$$\therefore \underline{w} = (G^T G)^{-1} G^T \underline{y} = G^+ \underline{y}$$

- e G^+ é chamada de matriz pseudo-inversa da matriz G .
- No entanto, para N grande, G pode se tornar mal condicionada (próxima da singularidade), dificultando a obtenção da matriz pseudo-inversa. Nestes casos, é necessária a obtenção iterativa da solução, através de um Filtro de Kalman ou outro método numérico adequado.