

Principais arquitecturas de redes

■ Aprendizagem Supervisionada

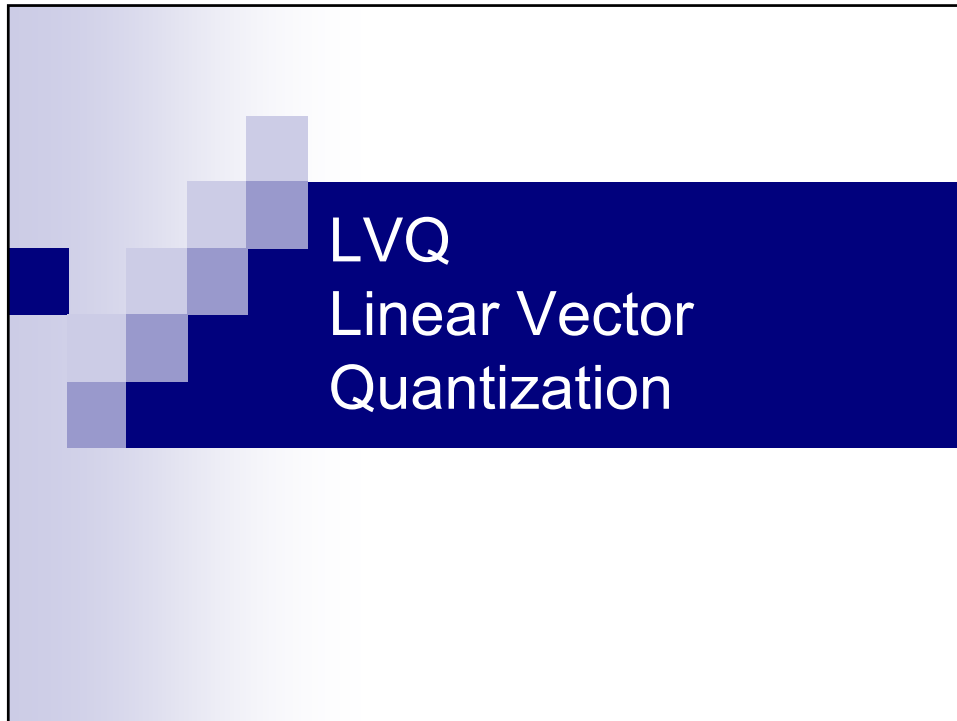
□ MLP com BP

- Multi-Layer Perceptron (perceptrão multicamada)
- Treinado com Backpropagation (retropropagação do erro)
- Variantes, vantagens, desvantagens, características...

■ Aprendizagem Não Supervisionada

□ SOM

- Self-Organizing Map (mapa auto-organizado)
- Variantes, vantagens, desvantagens, características...



LVQ – Linear vector quantization

- Derivado do SOM, mas aplicado a aprendizagem supervisionada
- Neurónios recebem classes “à partida”
- Actualização do neurónio depende da classe do neurónio e da classe do novo exemplo
 - Se a classe é a mesma, o neurónio é atraído
 - Se a classe é diferente, o neurónio é repellido

Algoritmo de treino do LVQ

■ Para cada padrão de entrada:

- 1) **Calcular** a distância entre o padrão de dados e todos os neurónios:

$$(d_{ij} = \|x_k - w_{ij}\|)$$

- 2) **Escolher** o neurónio vencedor

$$w_{winner} (w_{ij} : d_{ij} = \min(d_{mn}))$$

- 3) **Actualizar** cada neurónio de acordo com a regra

Para neurónios com a mesma classe do padrão

$$w_{ij} = w_{ij} + \alpha \cdot h(w_{winner}, w_{ij}) \cdot \|x_k - w_{ij}\|$$

Para neurónios com uma classe diferente

$$w_{ij} = w_{ij} - \alpha \cdot h(w_{winner}, w_{ij}) \cdot \|x_k - w_{ij}\|$$

- 4) **Repetir** o processo até que um critério de paragem seja atingido.

LVQ

■ Inicialização do mapa

- É comum fazer um SOM primeiro

■ Aprendizagem para cada novo exemplo

Para neurónios com a mesma classe do padrão

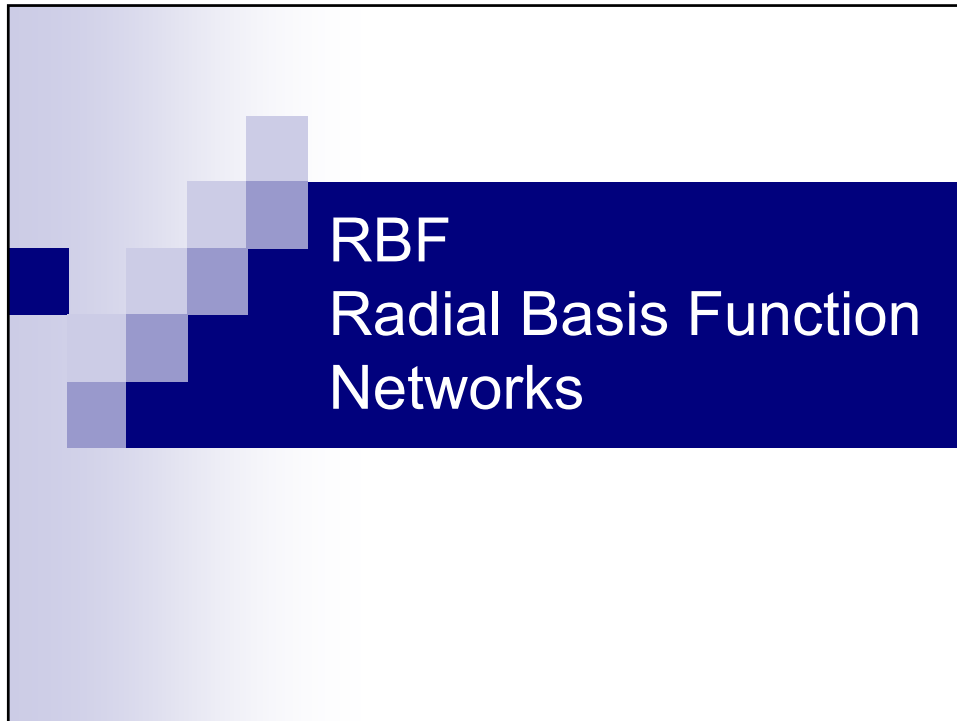
$$w_{ij} = w_{ij} + \alpha \cdot h(w_{winner}, w_{ij}) \cdot \|x_k - w_{ij}\|$$

Para neurónios com uma classe diferente

$$w_{ij} = w_{ij} - \alpha \cdot h(w_{winner}, w_{ij}) \cdot \|x_k - w_{ij}\|$$

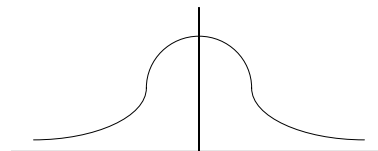
■ Resultado final

- Cada neurónio vai mapear uma só classe
- Maior separação entre classes



RBF – Radial Basis Function Networks

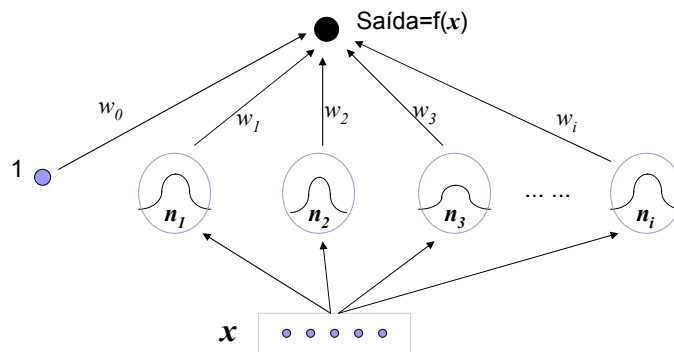
- Radial Basis Functions – Funções de base radial
 - O valor da função depende da distância a uma referência, e assimptoticamente decresce com esta
 - Também chamadas “kernel functions”
 - Parametrizadas pela posição central, largura, e eventualmente forma



Topologia de uma RBF

- Função implementada

- $f(x) = w_0 + \sum w_i K_i(d(n_i, x))$



Comparação MLP/RBF

- RBF só tem 1 camada escondida
- Cada neurónio num RBF é um “detector”, que define um “circulo” no espaço de entrada. Cada neurónio num MLP define um plano de separação.
- MLP usa produtos internos, RBF usa distâncias.
- MLP generaliza mais facilmente para zonas do espaço de entrada onde não há dados de treino. (RBF é uma aprendizagem mais localizada)

Implementação das RBF em SAS

■ Objecto comum às MLP

□ Escolher na opção “Architecture” RBF

- Equal Width – Todos os neurónios têm a mesma função de activação, e só a colocação dos neurónios no espaço é aprendida
- Unequal Width – O raio da cada RBF também é aprendido

□ Em opções avançadas

- Escolher função de activação
 - Vários tipos de funções possíveis

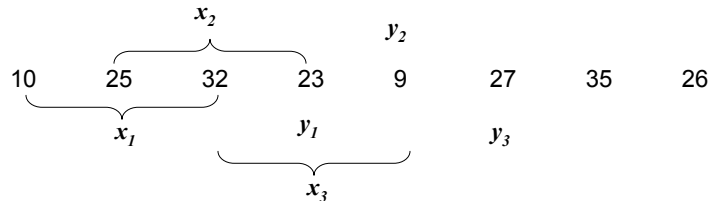
Redes para séries
temporais

Sistemas de Apoio à Decisão – Outras redes neuronais

V 1.0, V.Lobo, EN/ISEGI, 2005

Tempo “embitido” (embedded)

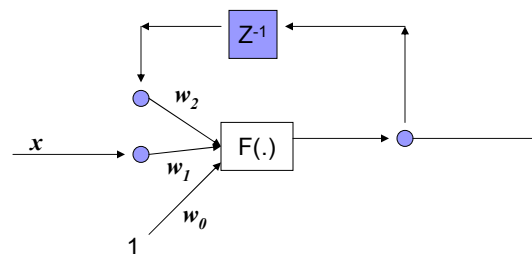
- Dividir a série temporal em vectores de n valores (atributos) + 1 valor alvo



- Usar uma transformação para outro espaço (por exemplo, transformada de Fourier ou Wavelets)

Redes recorrentes

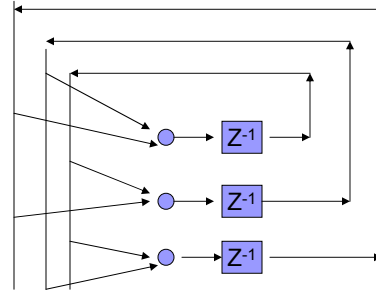
- Usam malhas de atraso na própria rede



- NARX – Non-linear Auto-Regressive with eXogenous inputs

Redes de Hopfield

- Para treinar
 - Forçar um padrão, e minimizar erros
- Para “associar”
 - Dar parte de um padrão, e a rede converge para um estado onde o resto do padrão é definido
- Trajectórias no espaço de pesos



Aprendizagem Hebbiana

- Se dois neurónios são activados ao mesmo tempo, as ligações entre eles deverão ser reforçadas
- Aprendizagem
 - Reforçar/enfraquecer ligações
- Utilização
 - Dar parte do padrão e associar o resto

