

2. Redes Neurais Artificiais

Prof. Renato Tinós

Depto. de Computação e Matemática (FFCLRP/USP)

2.4. Outras Redes Neurais Artificiais

2.4.1. Redes RBF

2.4.2. Mapas Auto-Organizáveis de Kohonen

2.4.3. Processamento Temporal

2.4.3.1. Rede Alimentada Adiante Focada Atrasada no Tempo (TLFN Focada)

2.4.3.2. Arquiteturas Recorrentes

- Rede de Elman
- Rede de Jordan
- Modelo geral de Redes Recorrentes
- Echo State Network (ESN)

2.4.3.3. Modelo de Hopfield

2.4.4. Projeto de RNAs

2.4.1. Redes RBF

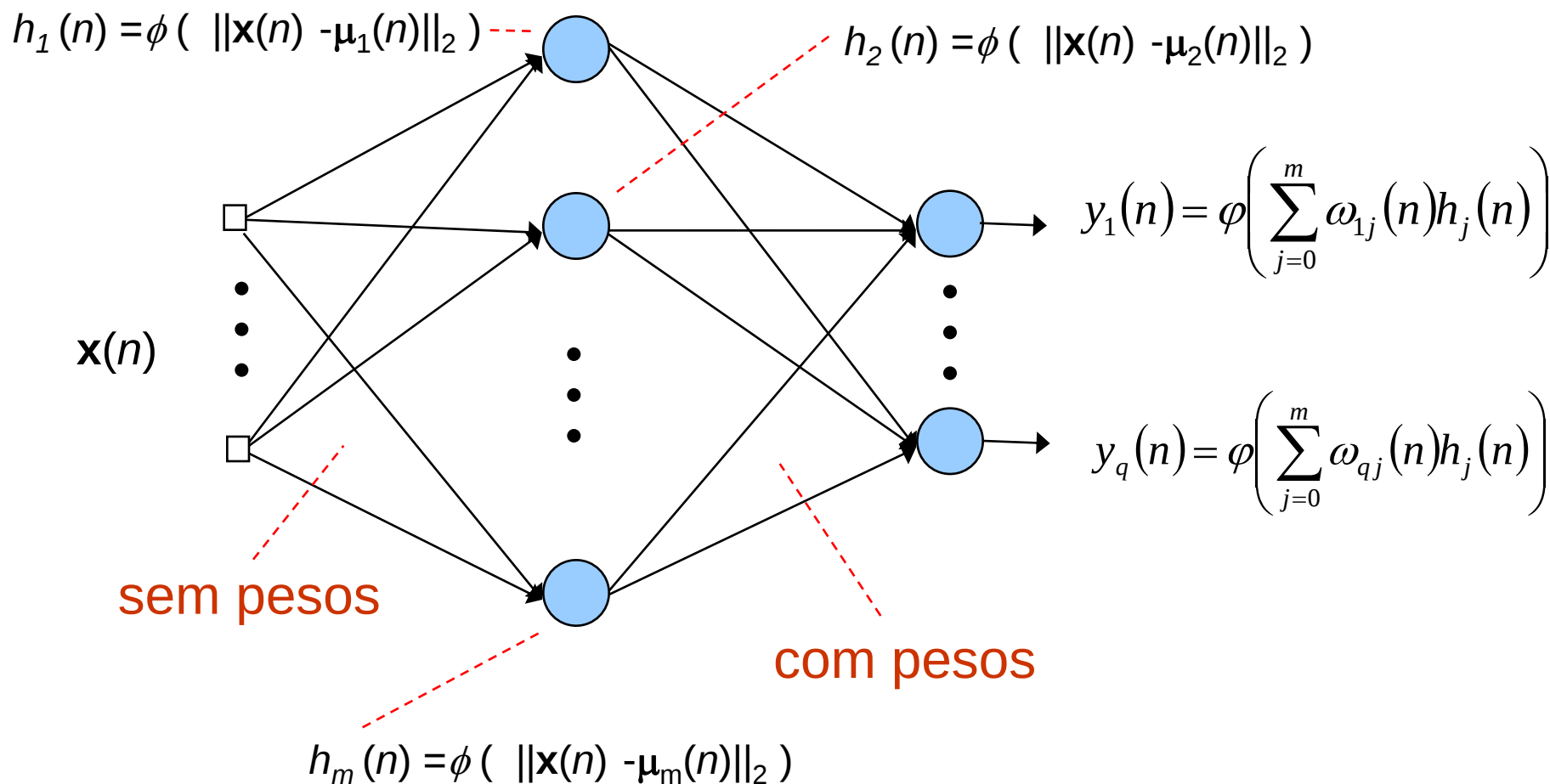
- **Tradicionalmente, são redes de duas camadas**
 - **Camada oculta**
 - » Utiliza funções de ativação não lineares (funções de base radial)
 - » Sem pesos (entre a camada de entrada e a camada oculta)
 - **Camada de saída**
 - » Utiliza funções de ativação tradicionais
 - Geralmente lineares
 - » Com pesos (entre a camada oculta e a camada de saída)

2.4.1. Redes RBF

camada de
entrada

camada oculta

camada de
saída



2.4.1. Redes RBF

- Cada neurônio da camada intermediária (**unidade radial**) computa a saída de uma função de base radial que possui dois parâmetros
 - Centro
 - » Vetor indicando o centro da unidade radial (μ_j)
 - Raio
 - » Indica o tamanho da área de influência da unidade radial
 - Geralmente utiliza-se áreas com um único parâmetro ($\rho_j = \rho$)

2.4.1. Redes RBF

- **Unidade radial**

- Saída da unidade radial j na iteração n

$$h_j(n) = \phi (\| \mathbf{x}(n) - \boldsymbol{\mu}_j(n) \|_2) \quad (2.4.1.1)$$

na qual a norma Euclidiana é dada por

$$\| \mathbf{x}(n) - \boldsymbol{\mu}_j(n) \|_2 = \sqrt{\sum_{i=1}^N (x_i(n) - \mu_i(n))^2} \quad (2.4.1.2)$$

- **Unidade radial**

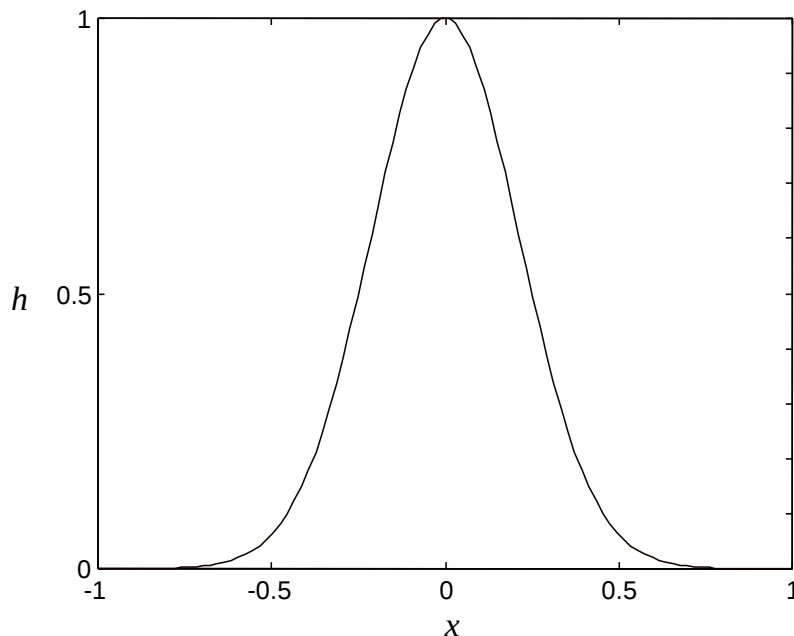
- Função de ativação das unidades radiais
 - » Não-linear
 - » Valor da saída varia com relação à distância entre o vetor de entrada e o centro da unidade radial
 - Fator de variação é dado pelo raio da unidade radial
 - » Existem várias funções radiais

2.4.1. Redes RBF

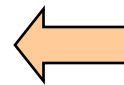
- **Unidade radial**

- Função de ativação das unidades radiais

- » Tipo mais comum: função Gaussiana



$$h_j(n) = e^{\left(-\frac{\|\mathbf{x}(n) - \boldsymbol{\mu}_j(n)\|_2^2}{2\rho^2} \right)} \quad (2.4.1.3)$$



Exemplo: Resposta da função Gaussiana com o centro em 0 ($\mu=0$) e $\rho=0,3$. Note que a ativação máxima ocorre em $x=\mu$.

2.4.1. Redes RBF

- **Camada de saída**

- Composta por neurônios com saídas dadas por

$$y_k(n) = \varphi \left(\sum_{j=0}^m \omega_{kj}(n) h_j(n) \right) \quad (2.4.1.4)$$

- Geralmente, as funções de ativação nesta camada são lineares

» Exemplo

$$y_k(n) = \sum_{j=0}^m \omega_{kj}(n) h_j(n) \quad (2.4.1.5)$$

2.4.1. Redes RBF

- **Classificação**

- MLP utiliza hiperplanos para gerar as FDs

- » Definidos por funções da forma $f(\mathbf{w}^T \mathbf{x}) = 0$

- » Quando o conjunto de padrões de treinamento não é significativo, as FDs não são intuitivas e robustas

- ❑ Definição das regiões de decisão é arbitrária em regiões do espaço de entradas não ocupadas por padrões de treinamento

- ❑ FDs poderiam ser colocadas em posições mais conservadoras

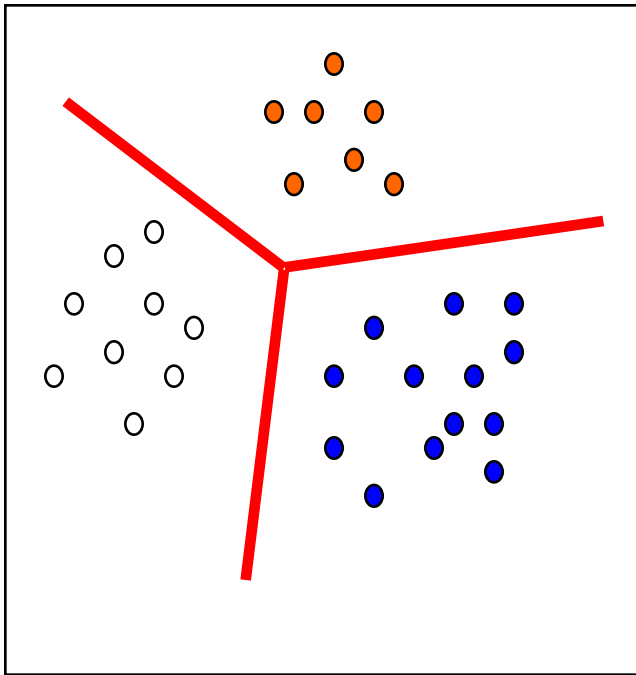
- RBF utiliza hiperelipsóides para gerar as FDs

- » Definidos por funções da forma $\phi(\|\mathbf{x} - \boldsymbol{\mu}_j\|_2)$

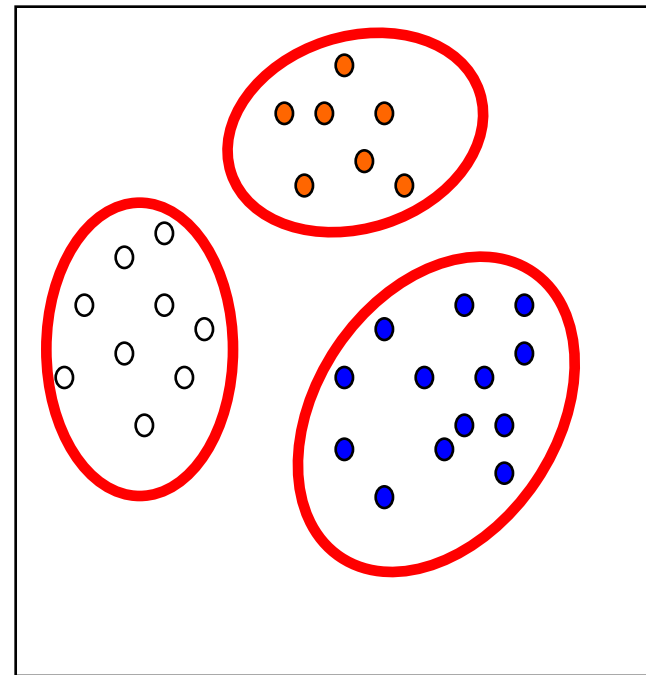
- ❑ Distância Euclidiana entre o vetor de entrada e o vetor que define o centro da unidade radial (neurônio da camada escondida)

2.4.1. Redes RBF

■ Exemplo



MLP



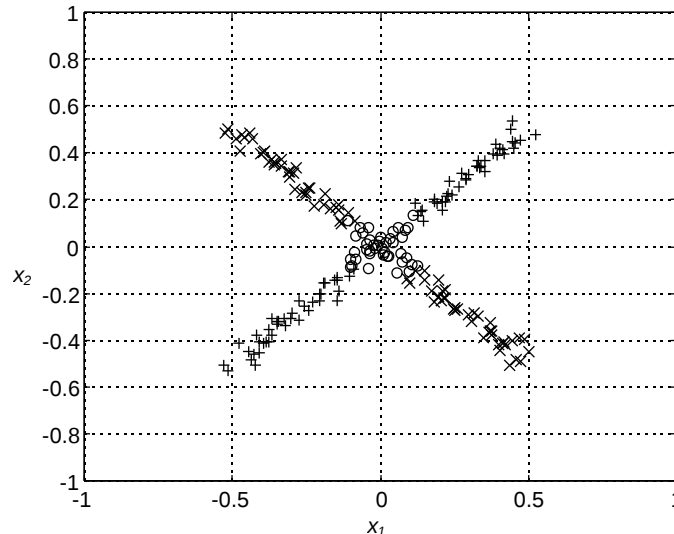
Rede RBF

2.4.1. Redes RBF

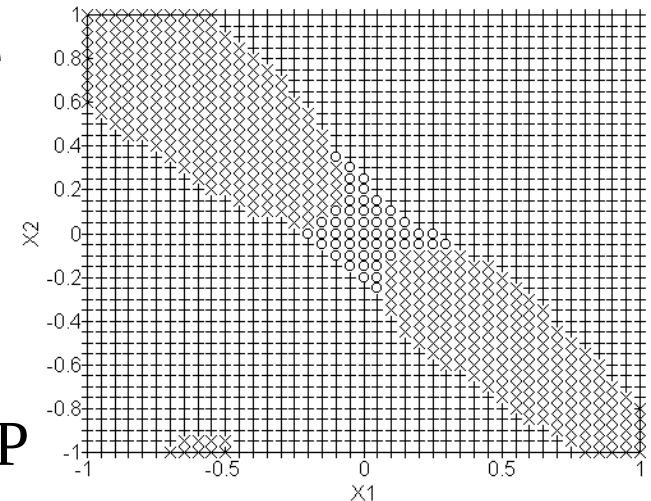
- **Fronteiras de decisão**

- São, em geral, mais robustas e intuitivas do que no MLP já que utiliza as distâncias entre os centros das unidades radiais e os exemplos a serem classificados

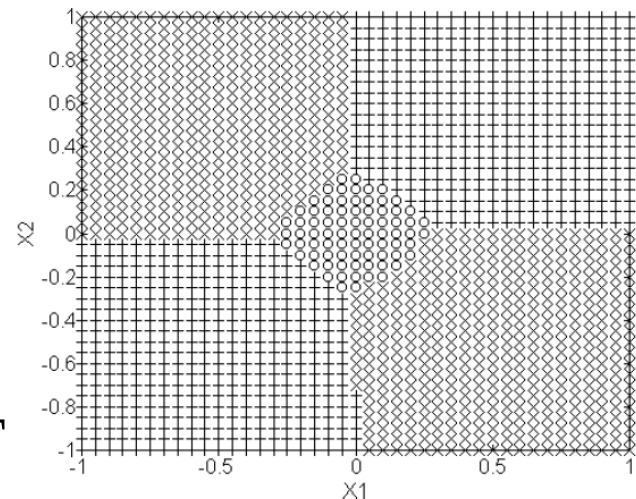
» Exemplo: espaço de classificação gerado pelo MLP



MLP



RBF



2.4.1. Redes RBF

- **Aproximação de Funções**

- **Projeção Linear:** combinação linear de funções elementares (bases)

$$\hat{f}(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^p w_i \varphi_i(\mathbf{x}) \quad (2.4.1.6)$$

sendo $\mathbf{w}$ um vetor de números reais escolhido de tal forma que, para um valor real ε suficientemente pequeno,

$$\left| f(\mathbf{x}) - \hat{f}(\mathbf{x}, \mathbf{w}) \right| < \varepsilon$$

2.4.1. Redes RBF

- **Aproximação de Funções**

- Repare que a Eq. (2.4.1.5) é igual a eq. da saída da Rede RBF com a camada de saída com ativação linear, ou seja,

sendo

$$y_k(n) = \sum_{j=0}^m \omega_{kj}(n) h_j(n) \quad (2.4.1.7)$$

$$h_j(n) = \phi (\| \mathbf{x}(n) - \boldsymbol{\mu}_j(n) \|_2) \quad (2.4.1.8)$$

2.4.1. Redes RBF

- **Projeção linear para o caso da rede RBF**
 - As bases (Eq. 2.52) são as funções radiais
 - Utiliza um conjunto de bases adaptativas
 - » As bases não são pré-determinadas
 - Como na Análise de Fourier ou nas Wavelets
 - » Bases são determinadas a partir do conjunto de dados entrada/saída (treinamento)
 - » Bases dependem dos parâmetros das unidades radiais e das entradas
 - » Pesos da camada de saída são ajustados para achar a melhor projeção da saída
 - » Treinamento é difícil porque deve-se achar as melhores
 - Bases (parâmetros das unidades radiais)
 - Projeções (pesos da camada de saída)

2.4.1. Redes RBF

- **Treinamento**

- O treinamento se resume a encontrar os seguintes parâmetros livres:
 - » Centros e raios das unidades radiais
 - Geralmente os raios são considerados fixos
 - » Vetor de pesos (incluindo o *bias*) entre a camada oculta e a camada de saída

2.4.1. Redes RBF

- **Treinamento**

- Geralmente, o treinamento é feito em dois estágios,

- » Primeiro estágio

- ☐ Considerando-se os raios das unidades radiais fixos, devem ser achados os centros das unidades radiais

- » Segundo estágio

- ☐ Conhecidos as ativações das unidades radiais para o conjunto de treinamento, deve-se determinar os pesos da camada de saída

2.4.1. Redes RBF

- **Treinamento**

- Primeiro estágio

- » Na abordagem mais simples, todos os padrões de treinamento são escolhidos como centros de unidades radiais

- similar ao método IBL

- se N padrões são empregados no treinamento, N unidades radiais são criadas

- Problema 1: o uso de muitas unidades radiais pode ocasionar lentidão de operação da rede RBF (alto custo computacional) e ineficiência na segunda parte do treinamento (segundo estágio)

- Problema 2: sobre treinamento ou *overfitting*

- » Número de unidades radiais depende da complexidade dos dados

- Número de aglomerados necessários para realizar a tarefa definida

2.4.1. Redes RBF

- **Treinamento**

- Primeiro estágio

- » Outras abordagens

- ☐ Seleção aleatória dos centros

- ☐ *Forward Selection*

- ☐ Métodos de Clusterização

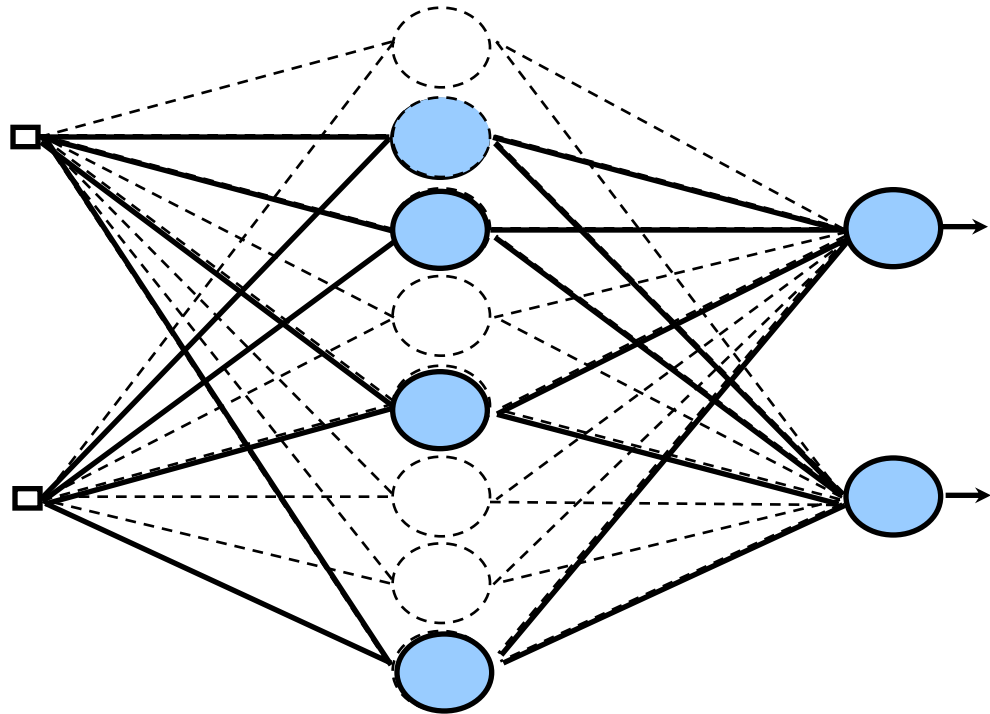
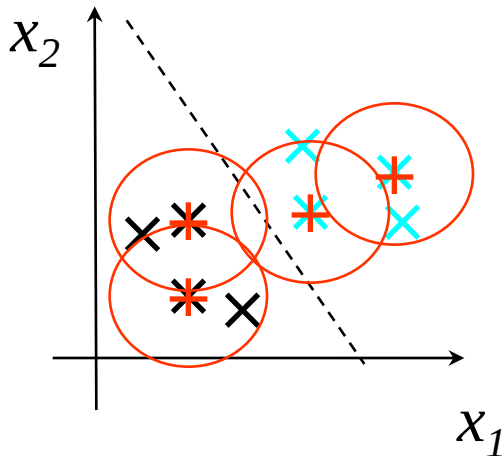
- K-means

- Mapa Auto-Organizável de Kohonen

- ☐ Algoritmos Genéticos

2.4.1. Redes RBF

- **Treinamento**
 - Primeiro estágio
 - » *Forward Selection*



2.4.1. Redes RBF

- **Treinamento**

- Segundo estágio

- » Determinação do vetor de pesos entre a camada oculta (das unidades radiais) e a camada de saída
 - » Se as unidades radiais já foram definidas (primeiro estágio), o problema da determinação do vetor pesos é supervisionado
 - » Se, ainda, as funções de ativações dos neurônios da camada de saída são lineares, o vetor de pesos pode ser obtido de maneira direta minimizando-se a soma dos erros quadráticos

2.4.1. Redes RBF

- **Treinamento**

- Segundo estágio

» Usando o Método dos Mínimos Quadráticos (LMS), o vetor de pesos é dado por

$$\mathbf{W} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{D} \quad (2.4.1.9)$$

$$\mathbf{W} = [\mathbf{w}_1 \quad \mathbf{w}_2 \quad \dots \quad \mathbf{w}_q]$$

na qual

$$\mathbf{w}_k = [\mathbf{w}_{k1} \quad \mathbf{w}_{k2} \quad \dots \quad \mathbf{w}_{km}]^T$$

$$\mathbf{D} = [\mathbf{d}_1 \quad \mathbf{d}_2 \quad \dots \quad \mathbf{d}_q]$$

$$\mathbf{d}_k = [\mathbf{d}_k(1) \quad \mathbf{d}_k(2) \quad \dots \quad \mathbf{d}_k(N)]^T$$

$$\mathbf{H} = \begin{bmatrix} h_1(1) & h_2(1) & \dots & h_m(1) \\ h_1(2) & h_2(2) & \dots & h_m(2) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(N) & h_2(N) & \dots & h_m(N) \end{bmatrix}$$

2.4.2. Mapas Auto-Organizáveis de Kohonen

- Em várias aplicações de RNAs, é desejável que a rede decida, através de um processo de auto-organização, quais são as classes em um problema de classificação
 - Agrupamento
- Para isso é necessário que:
 - Padrões pertencentes a mesmo agrupamento possuam semelhanças
 - A rede consiga identificar estas semelhanças
 - » Aprenda um critério para agrupar os dados

2.4.2. Mapas Auto-Organizáveis de Kohonen

- **Propósito de RNA que utiliza auto-organização é**
 - Descobrir padrões ou características significativas nos dados de entrada
 - » Sem informações sobre a qualidade das soluções ou saídas desejadas
 - Agrupar dados em aglomerados (*clusters*)
- **Para isso, o algoritmo de treinamento utiliza um conjunto de regras de natureza local**

2.4.2. Mapas Auto-Organizáveis de Kohonen

- **Redes auto-organizáveis são mais semelhantes às estruturas neurobiológicas que as redes supervisionadas**
 - Determinadas áreas do cérebro são responsáveis por funções específicas
 - » Fala
 - » Visão
 - » Controle de movimentos
 - Cada área pode conter sub-áreas

2.4.2. Mapas Auto-Organizáveis de Kohonen

- **Redes SOM (*Self-Organizing Maps*)**
 - Propostas por Teuvo Kohonen
 - Também chamadas de Redes de Kohonen
 - Criam mapas topológicos a partir dos padrões de treinamento
 - » Padrões semelhantes ativam regiões próximas do mapa

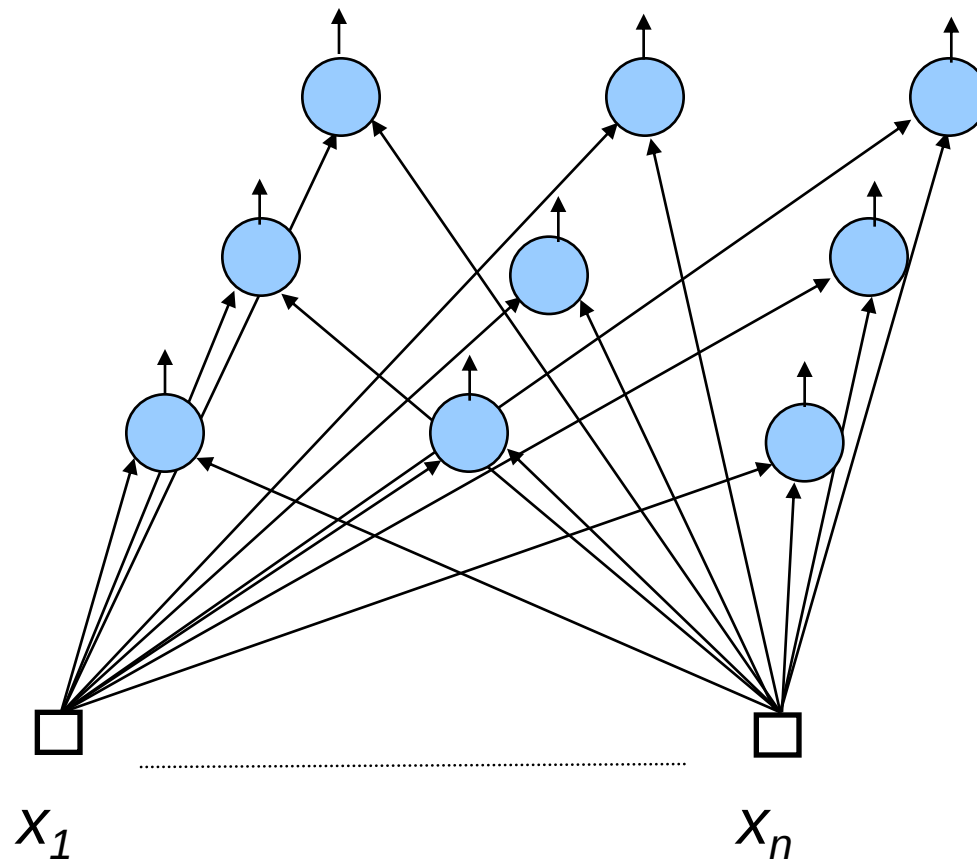
2.4.2. Mapas Auto-Organizáveis de Kohonen

- **Utilizam algoritmo de aprendizado baseado em conceitos de auto-organização**
- **Baseadas no mapeamento realizado pelo cérebro**
 - Permite representação de dados n -dimensionais em um espaço c -dimensional ($c \ll n$)
 - Utiliza técnica de quantização de vetores para comprimir dados dos vetores de entrada

2.4.2. Mapas Auto-Organizáveis de Kohonen

- **Arquitetura**
 - Geralmente uma camada bi-direcional
 - » Grade plana (reticulado)
 - Cada neurônio
 - » Recebe todas as entradas e gera saída
 - Pode ser utilizada uma hierarquia de camadas

2.4.2. Mapas Auto-Organizáveis de Kohonen



2.4.2. Mapas Auto-Organizáveis de Kohonen

- **Estados de ativação**
 - $[1, m]$
- **Função de ativação**
 - Baseada em distância Euclidiana

$$d_j = \|\mathbf{x}(t) - \mathbf{w}_j(t)\|_2 \quad (2.4.2.1)$$

- **Neurônio vencedor i**

$$i(\mathbf{x}(t)) = \arg \min_{j=1, \dots, m} \|\mathbf{x}(t) - \mathbf{w}_j(t)\|_2 \quad (2.4.2.2)$$

2.4.2. Mapas Auto-Organizáveis de Kohonen

- **Treinamento**

- Atualização dos pesos

- » Atualiza neurônio vencedor e seus vizinhos dentro de uma certa vizinhança
 - » Vizinhança e taxa de aprendizado convergem para zero durante o treinamento
 - » Cria regiões que respondem a um grupo de entradas semelhantes

2.4.2. Mapas Auto-Organizáveis de Kohonen

- **Treinamento**

- Atualização dos pesos

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \eta(t) h_{ji}(t) (\mathbf{x}(t) - \mathbf{w}_j(t)) \quad (2.4.2.3)$$

- » $h_{ji}(t)$: define vizinhança

- » $\eta(t)$: taxa de aprendizado

2.4.2. Mapas Auto-Organizáveis de Kohonen

- 1. Iniciar conexões com pequenos valores aleatórios;**
- 2. Definir vizinhança e taxa de aprendizado iniciais**
- 3. Repita**

Para cada padrão de entrada x

Para cada nodo j

Calcular d_j

Selecionar nodo i com menor d

Atualizar pesos de i e seus vizinhos

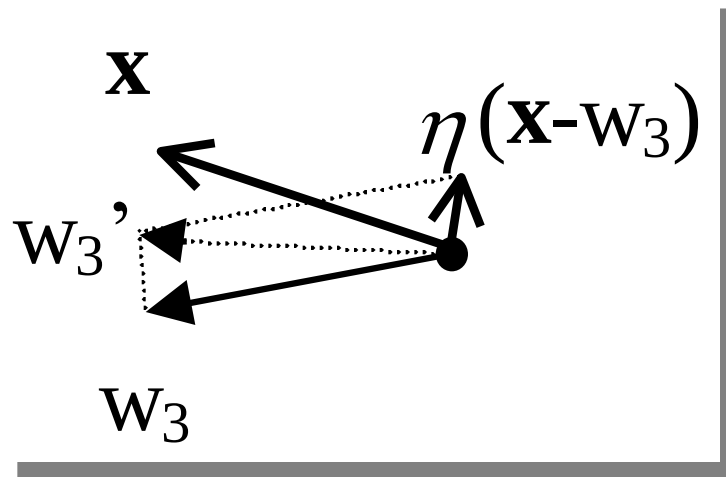
Reduzir taxa de aprendizado η

Reduzir vizinhança h_{ji}

Até que critério de parada seja satisfeito

2.4.2. Mapas Auto-Organizáveis de Kohonen

- **Adaptação dos pesos**
 - O nodo vencedor (nodo 3 para o exemplo) atualiza seus pesos de forma a se aproximar mais ainda do vetor de entrada $\mathbf{x}$:



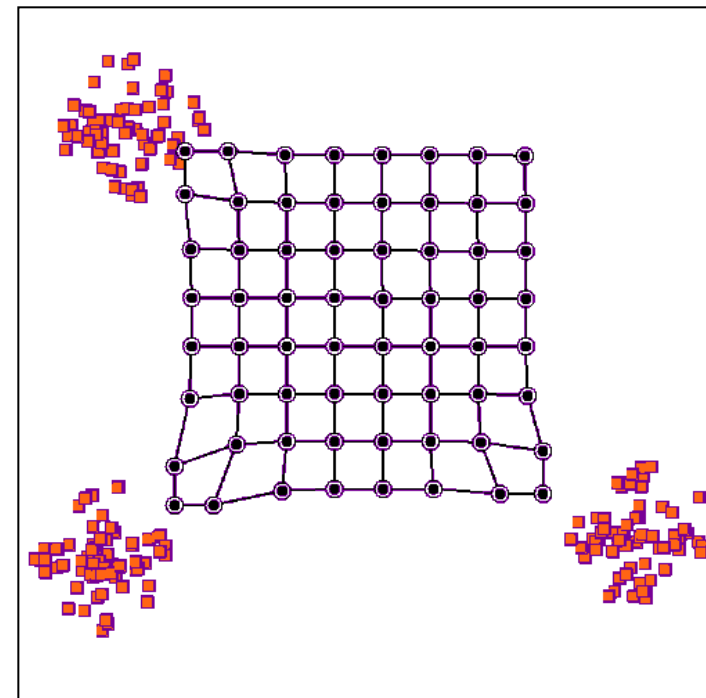
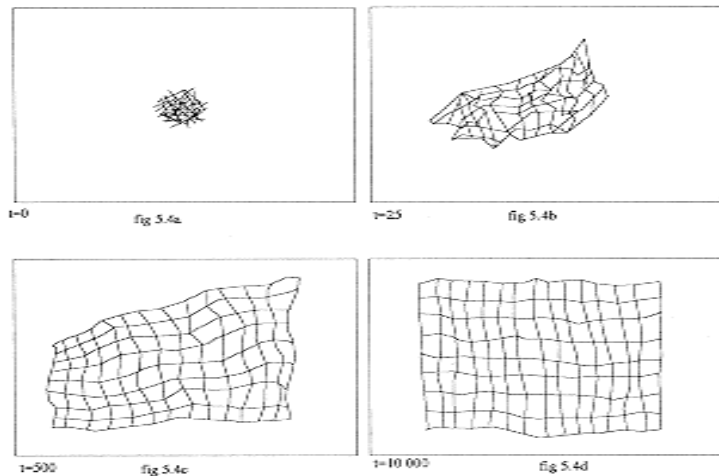
2.4.2. Mapas Auto-Organizáveis de Kohonen

- **Vizinhança (h)**

- Define quantos nós em torno do nó vencedor terão seus pesos ajustados
- Tamanho modificado dinamicamente durante treinamento
 - » Inicialmente grande (ex. todos os nós)
 - » Reduzido progressivamente até limite pré-definido
 - Taxa de redução é geralmente linear com o número de ciclos
 - » Pode ter diferentes formatos
 - Quadrado
 - Hexágono
 - Círculo
 - Irregular

2.4.2. Mapas Auto-Organizáveis de Kohonen

- **Exemplo 1**
 - Alteração de pesos durante o treinamento



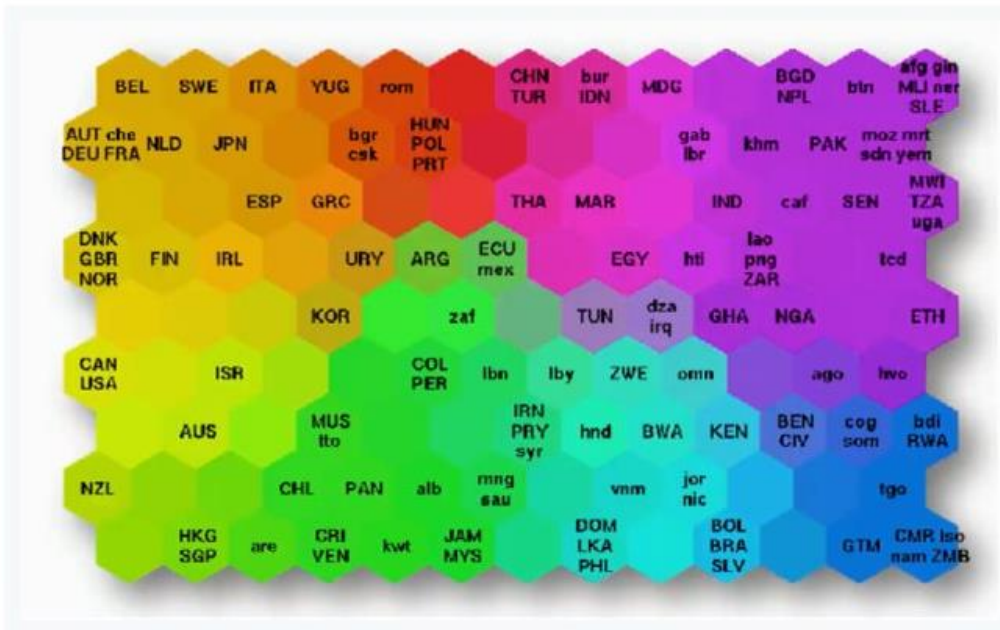
https://en.wikipedia.org/wiki/Self-organizing_map#/media/File:TrainSOM.gif

2.4.2. Mapas Auto-Organizáveis de Kohonen

- **Após treinamento, rede SOM forma agrupamentos**
 - Muito útil para visualização
 - » Redução de dimensionalidade
 - Se as classes forem conhecidas:
 - » Grupos podem ser rotulados para indicar classe que representam
 - » Permite classificação de padrões desconhecidos

2.4.2. Mapas Auto-Organizáveis de Kohonen

What you see below is an actual SOM. This map represents the levels of economic wellbeing across a wide range of countries.

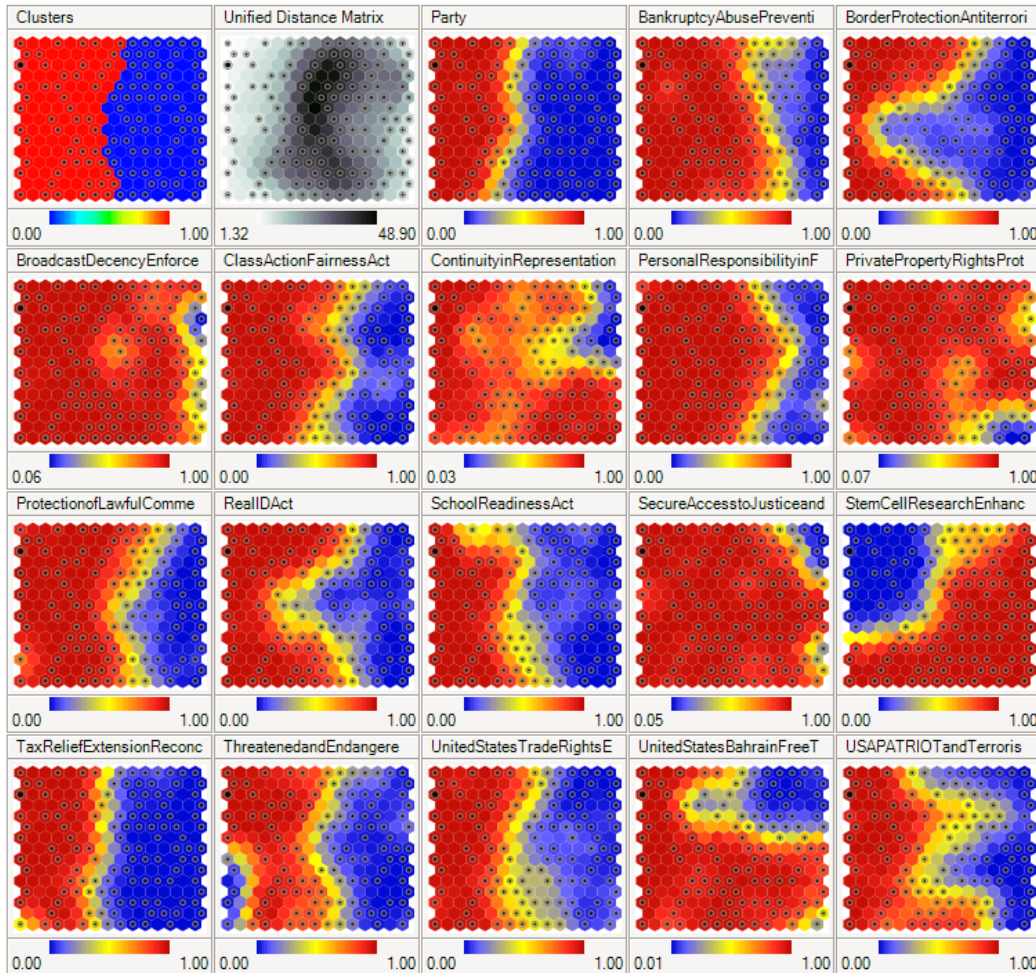


As you can see, the countries are lined up in a cluster, and their order inside that cluster is based on various indicators (e.g. health conditions, quality of education, average income per capita, etc.).

You can imagine this data set of 200+ countries to have started off with 39 different columns. It's tough to imagine such a bulk of data, isn't it?

<https://www.superdatascience.com/blogs/the-ultimate-guide-to-self-organizing-maps-soms>

2.4.2. Mapas Auto-Organizáveis de Kohonen



A self-organizing map showing [U.S. Congress](#) voting patterns. The input data was a table with a row for each member of Congress, and columns for certain votes containing each member's yes/no/abstain vote. The SOM algorithm arranged these members in a two-dimensional grid placing similar members closer together. **The first plot** shows the grouping when the data are split into two clusters. **The second plot** shows average distance to neighbours: larger distances are darker. **The third plot** predicts [Republican](#) (red) or [Democratic](#) (blue) party membership. **The other plots** each overlay the resulting map with predicted values on an input dimension: red means a predicted 'yes' vote on that bill, blue means a 'no' vote. The plot was created in [Synapse](#).

https://en.wikipedia.org/wiki/Self-organizing_map#/media/File:Synapse_Self-Organizing_Map.png

2.4.3. Processamento Temporal

- **Diversas aplicações requerem o processamento temporal de informações**

- **Ex.: Previsão de Séries Temporais**

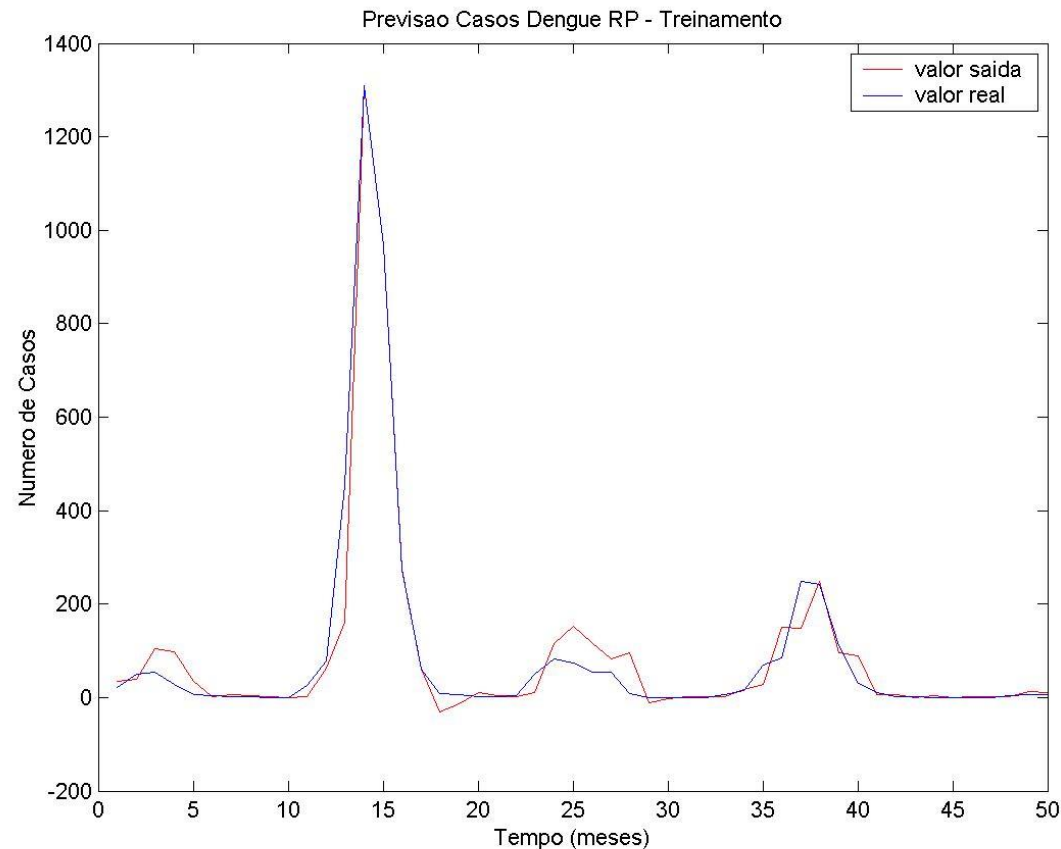
» dado um conjunto de exemplos

$$\{ \mathbf{x}(1) , \mathbf{x}(2) , \dots , \mathbf{x}(n) \}$$

prever a saída $\mathbf{x}(n+1)$

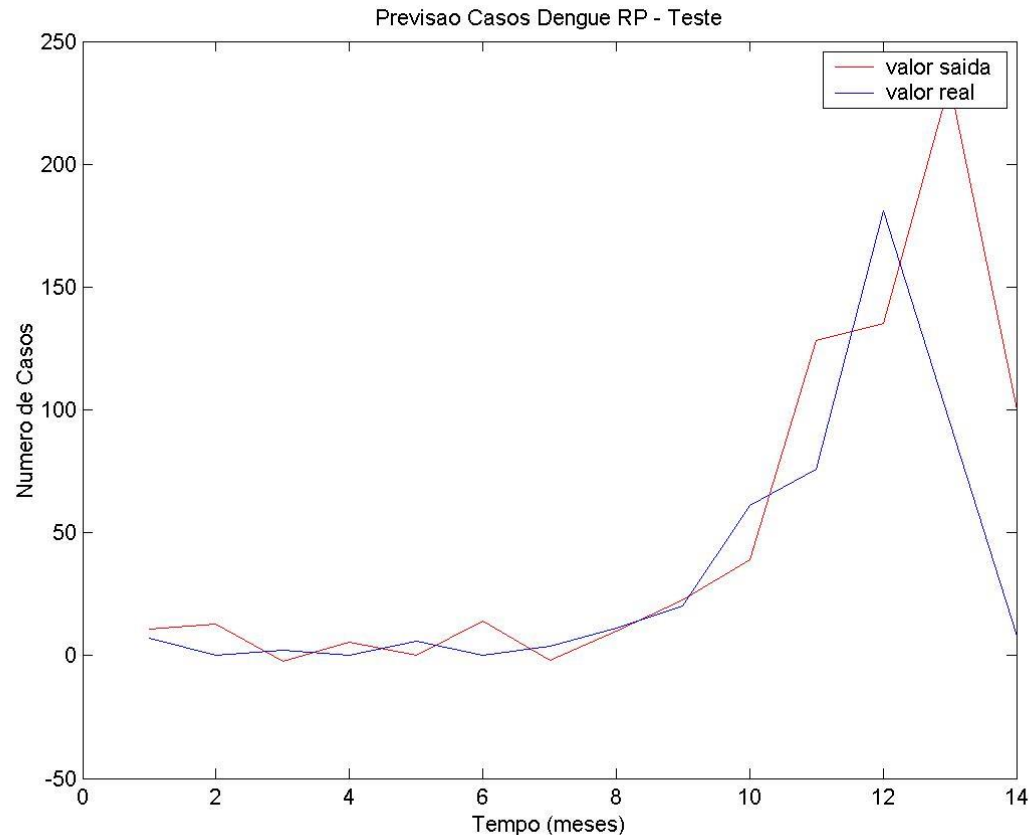
2.4.3. Processamento Temporal

- **Exemplo 6.2. Sistema de Auxílio à Previsão de Epidemias de Dengue**
 - Resultados



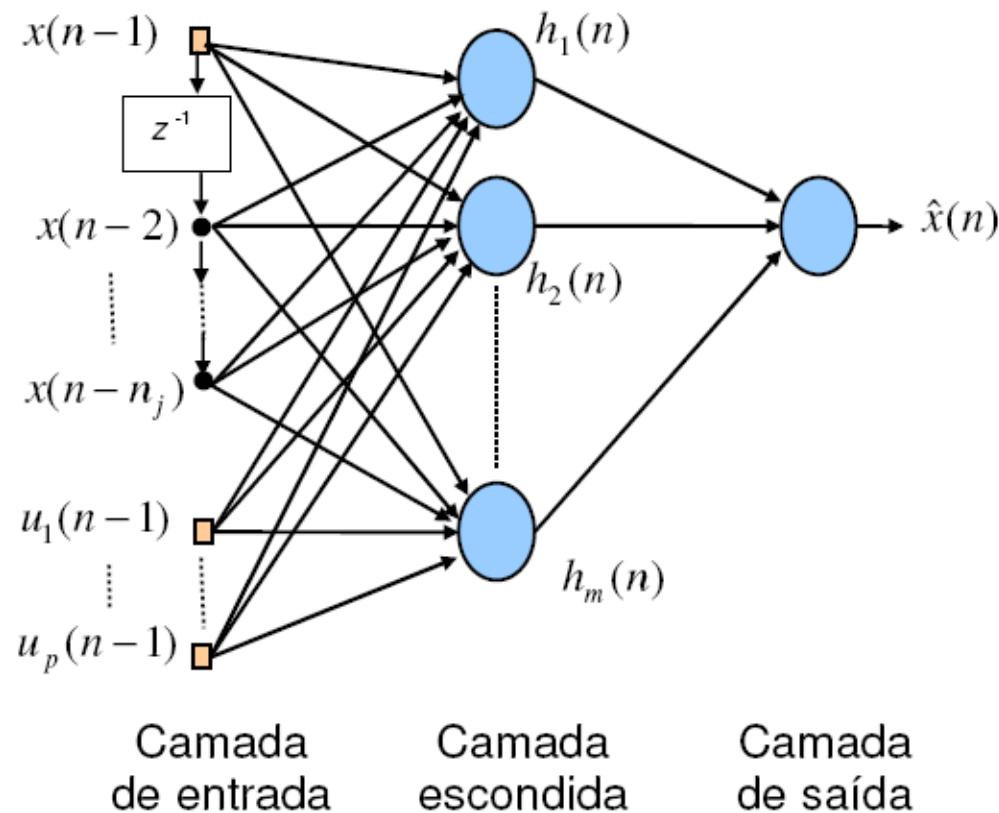
2.4.3. Processamento Temporal

- **Exemplo 6.2. Sistema de Auxílio à Previsão de Epidemias de Dengue**
 - Resultados



2.4.3.1. TLFN Focada

- **Rede Neural Alimentada Adiante Focada Atrasada no Tempo (TLFN focada)**



2.4.3.1. TLFN Focada

- **Propriedades**

- **Tempo**

- » Informação presente no conjunto de entrada
 - » Memória de curto prazo localizada externamente à rede

- **RNA estática**

- » Treinamento padrão

- **Número de entradas**

- » Qual deve ser o atraso considerado?

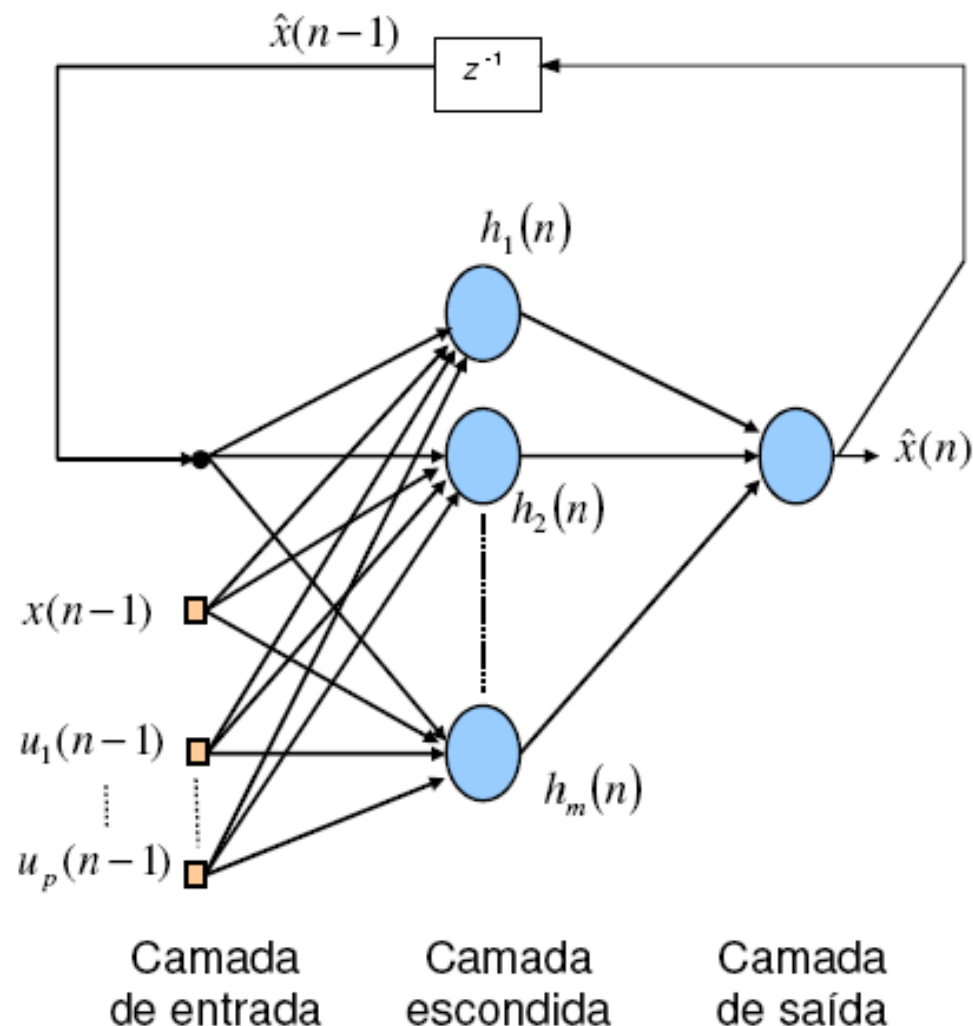
2.4.3.2. Redes Recorrentes

- **Laços de realimentação internos são considerados**
 - A informação dos neurônios de uma camada são realimentadas na mesma camada ou em camadas anteriores
- **Existem diversas topologias**

2.4.3.2. Redes Recorrentes

- **Rede de Jordan**

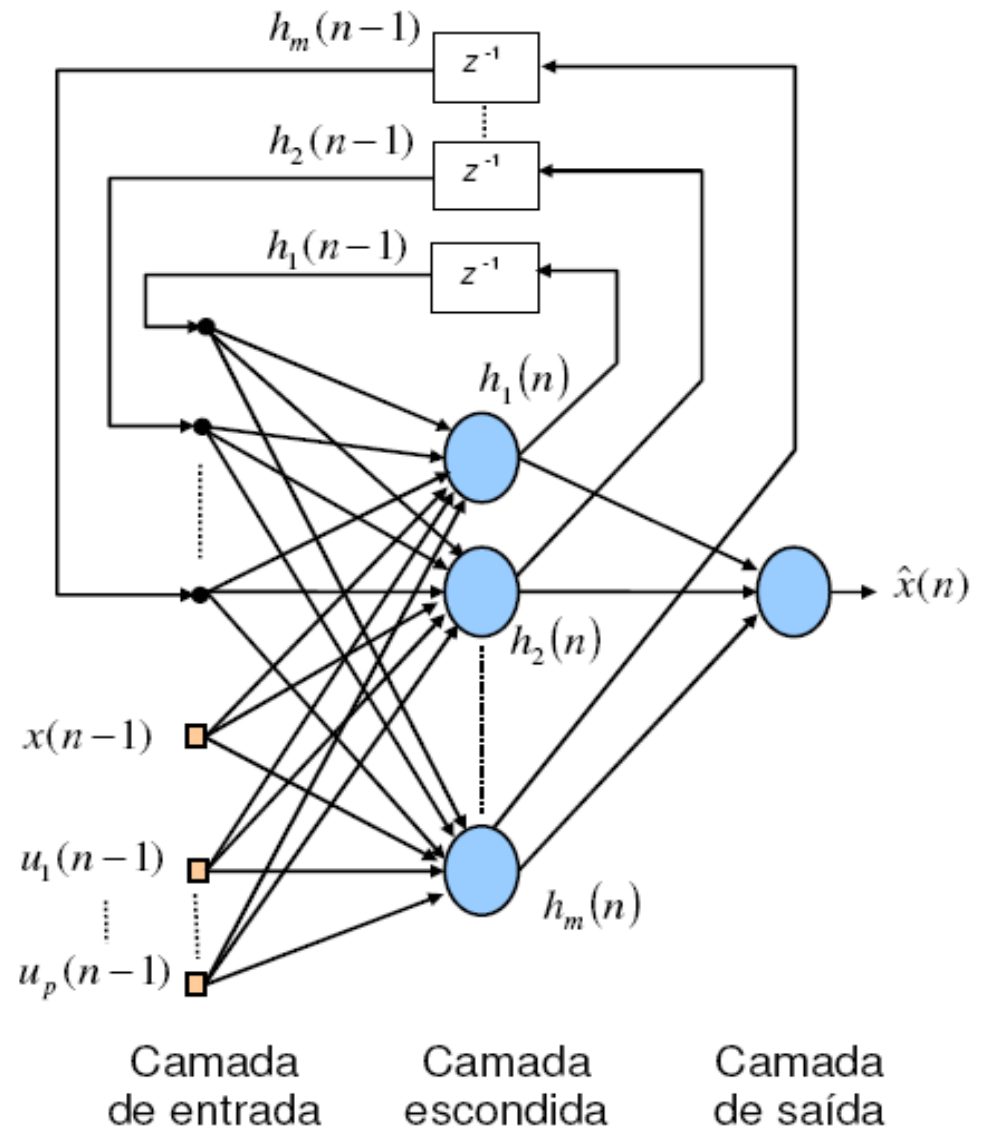
- Realimentação dos neurônios da camada de saída na unidade de contexto



2.4.3.2. Redes Recorrentes

- **Rede de Elman**

- Realimentação dos neurônios da camada oculta na unidade de contexto



2.4.3.2. Redes Recorrentes

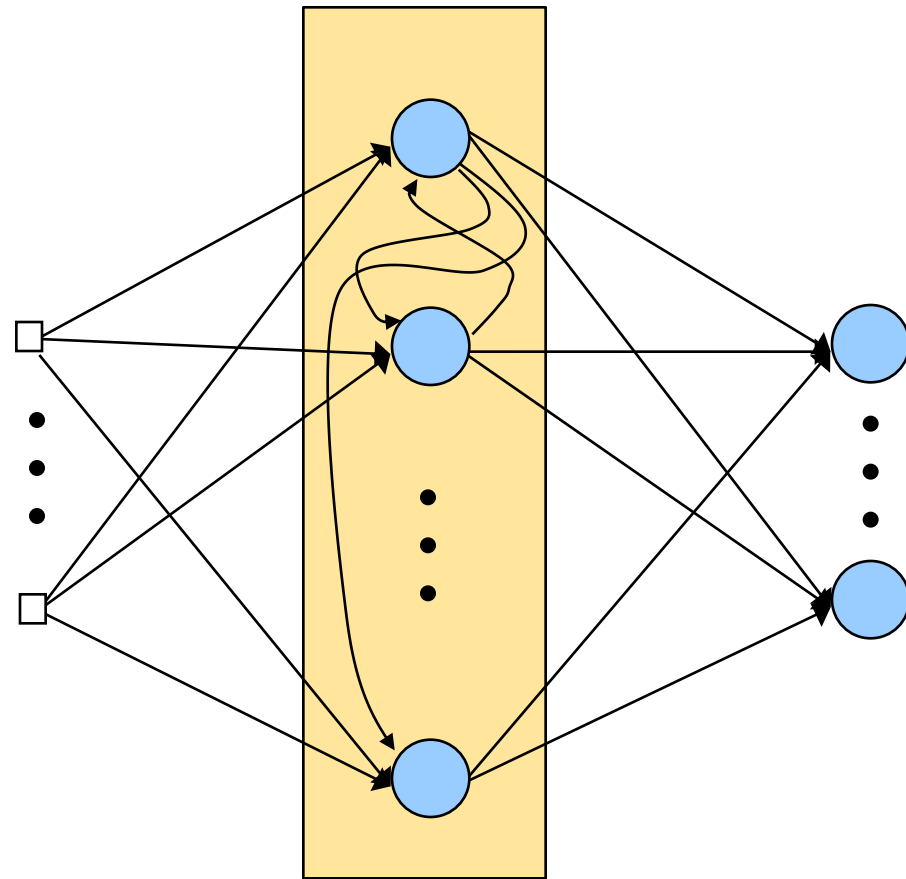
- **Modelo Geral**

- A informação de qualquer neurônio em qualquer camada pode ser realimentada para qualquer neurônio em qualquer camada
- Em geral, os algoritmos padrões de treinamento de redes estáticas não podem ser usados diretamente
 - » Algoritmos de treinamento especiais geralmente têm que ser desenvolvidos
 - Ex.: Retropropagação através do tempo

2.4.3.2. Redes Recorrentes

- **Echo State Network (ESN)**

- Os pesos do reservatório e da camada de entrada são aleatórios

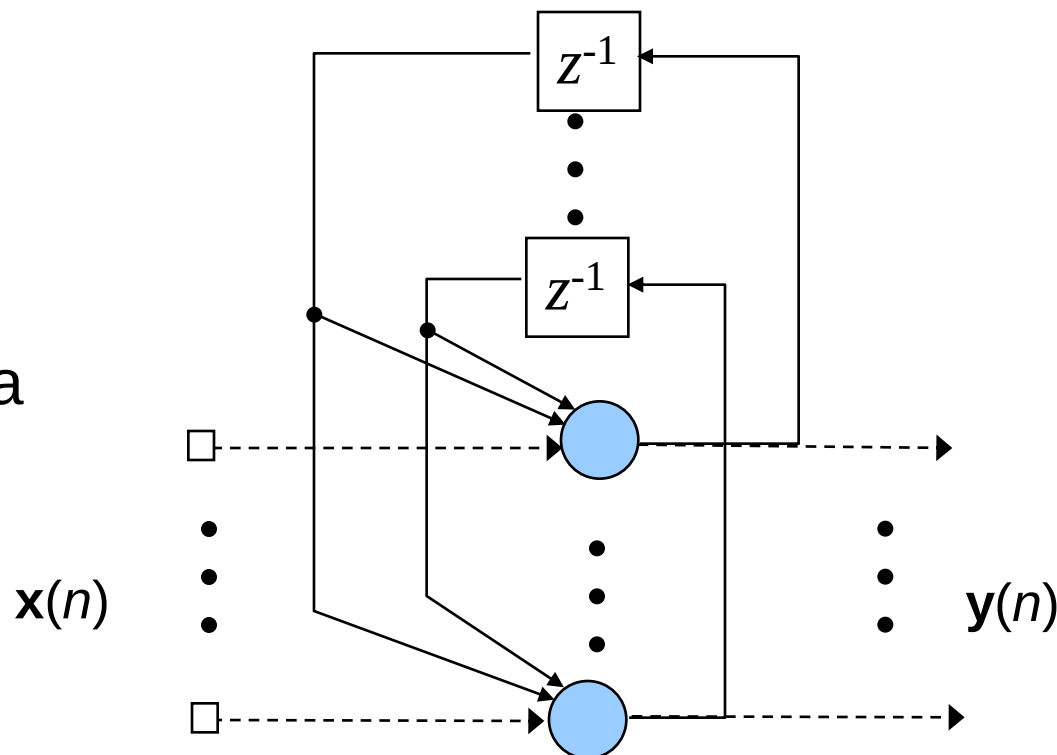


Reservatório

2.4.3.3. Modelo de Hopfield

- **Topologia**

- Sem unidades escondidas
- Matriz de pesos simétrica



2.4.3.3. Modelo de Hopfield

- **Rede de Hopfield Discreta**

- Saída do neurônio i ($i=1,\dots,m$)

$$y_i(n+1) = \text{sgn} \left(\sum_{j=1}^m \omega_{ij} y_j(n) - \omega_{i0} + x_i(n) \right) \quad (2.4.3.3.1)$$

sendo $\mathbf{x}$ um vetor binário que é válido somente na primeira iteração (condição inicial)

2.4.3.3. Modelo de Hopfield

- **Rede de Hopfield Discreta**

- Saída do neurônio i ($i=1,\dots,m$)

- » Após a apresentação do vetor $\mathbf{x}$, a rede relaxa naturalmente

- o termo correspondente à entrada x_i é removido da eq. anterior

- até que a saída estabilize

2.4.3.3. Modelo de Hopfield

- **Rede de Hopfield Discreta**

- Considere o problema de auto-associação de padrões
 - » Ou seja, dado o padrão $\mathbf{x}^p$, queremos que a saída estável seja $\mathbf{y}=\mathbf{x}^p$
- Para a saída estável i (considerando por simplicidade que o bias é igual a zero), a eq. (2.4.3.3.1) fica

$$y_i = \text{sgn} \left(\sum_{j=1}^m \omega_{ij} y_j \right) \quad (2.4.3.3.2)$$

- O ponto estável é chamado de **atrator**

2.4.3.3. Modelo de Hopfield

- **Rede de Hopfield Discreta**

- Hopfield mostrou que, neste caso, os pesos devem ser definidos por

$$\omega_{ij} \propto x_i^p x_j^p \quad (2.4.3.3.3)$$

- Ou seja, o vetor de pesos é definido automaticamente pelo padrão de entrada

2.4.3.3. Modelo de Hopfield

- **Rede de Hopfield Discreta**
 - O que é atrativo nas redes de Hopfield é que $y(n)$ irá convergir para x^p mesmo para uma entrada parcialmente completa ou corrompida por ruído.
 - Este sistema pode também armazenar múltiplos padrões, no entanto o número de padrões que podem ser armazenados é limitado.

- **Referências**

- Haykin, S. S.. *Redes neurais: princípios e prática*. 2ª ed., Bookman, 2001.
- Principe, J. C.; Euliano, N. R. & Lefebvre, W. C. *Neural and Adaptive Systems: Fundamentals Through Simulations*. John Wiley & Sons, Inc. 2000
- Braga, A.P.; Carvalho, A. C. P. L. F. & Ludermir, T.B.. *Redes neurais artificiais: Teoria e Aplicações*. LTC, 2000.