REDES NEURAIS EM ESTATÍSTICA

Basílio de Bragança Pereira
FM (Faculdade de Medicina), NESC (Núcleo de Estudos de Saúde Coletiva) e COPPE,UFRJ

&

Cleide Vital da Silva Rodrigues
PEP/COPPE,UFRJ

13° SINAPE - CAXAMBU
1998

CONTEÚDO

FUNDAMENTOS

APLICAÇÕES EM ESTATÍSTICA

FUNDAMENTOS

FUNDAMENTOS

INTRODUÇÃO

Durante vários anos a humanidade acreditou que todos os processos que constituem a inteligência já estivessem totalmente compreendidos. O avanço da tecnologia computacional proporcionou o desempenho cada vez melhor e mais rápido das tarefas complexas em comparação com o cérebro.

Os computadores modernos são inteiramente baseados em um processador central que adiciona e multiplica números binários. Todas as tarefas executadas pelos computadores, incluindo cálculos, processamento de palavras e controle robótico, são feitas através de operações binárias. A informação é armazenada nos registradores.

Recentemente, a indústria computacional fez grandes avanços fazendo processadores binários menores e mais rápidos para facilitar o desenvolvimento de novas aplicações na tecnologia.

Os computadores são rápidos em computação numérica, até excedendo a capacidade humana. Entretanto, o cérebro humano tem muitas habilidades que são desejadas em um computador como por exemplo: a habilidade de rapidamente identificar características, mesmo na presença de ruído; entender, interpretar; fazer inferências e julgamentos baseados em esperiências passadas e relacioná-las com situações que nunca foram encontradas antes; e, mesmo com uma área danificada não perder completamente a funcão. Assim, embora o computador seja mais rápido que o cérebro humano em computação numérica, o cérebro é melhor em outras tarefas. Esta é a motivação para tentar entender e modelar o cérebro humano.

O NEURÔNIO

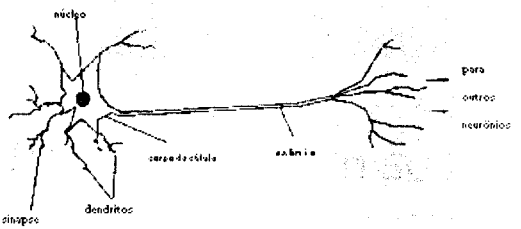
•O cérebro humano $\approx$ 10 bilhões de neurônios.

•Cada um constituído de uma *célula*, um *axônio* e uma árvore dendrítica formada por ramificações chamadas *dendritos*.

•As unidades que ligam os neurônios, isto é, conectam os dendritos aos axônios, são chamadas de *sinapses*.

•As *sinapses* constituem o mecanismo transmissor de informação entre os neurônios. A informação recebida pelo neurônio é processada e produz uma saída que pode excitar ou inibir outros neurônios.

•O tipo mais comum de *sinapse* é a sinapse química, que converte um sinal químico e retorna um sinal elétrico e vice-versa.



Neurônio real

A figura acima apresenta um neurônio biológico.

A capacidade do cérebro em realizar rapidamente determinadas tarefas complexas é devido, principalmente:

* A grande quantidade de neurônios.

* As conexões complexas entre os neurônios.

* Ao paralelismo.

* A plasticidade - é uma habilidade que permite a adaptação ou criação de novas conexões sinápticas entre neurônios e também a modificação de sinapses já existentes.

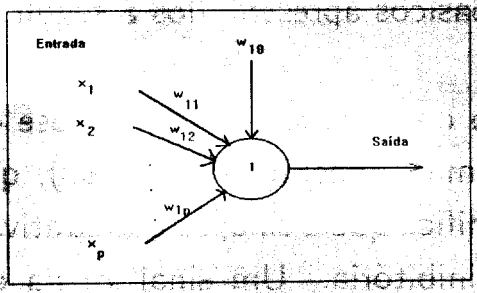
O QUE É REDE NEURAL ARTIFICIAL?

A *Rede Neural Artificial* (*RNA*) é uma estrutura que tem como modelo o sistema neural biológico.

•Origem das *RNA* - tem como marco o modelo de McCulloch e Pitts.

•Neurônio artificial - elemento básico do processamento de uma *RNA*. Tem característica similar ao neurônio biológico. O neurônio artificial tem entradas, que podem ser negativas (inibição) ou positivas (excitação) e que produzem uma única saída e depois torna-se *sinapse* com um ou mais processadores.

•A diferença entre uma *RNA* e um sistema computacional convencional é que as *RNA* têm vários processadores trabalhando ao mesmo tempo que servem simultaneamente como elementos de memória e de processamento.



Neurônio artificial

NEURÔNIO ARTIFICIAL

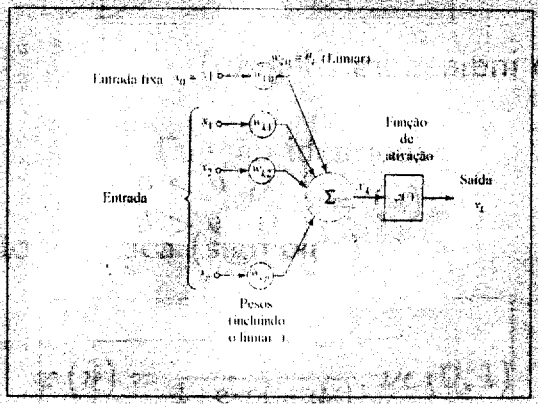
O neurônio artificial pode ser identificado pelos três elementos básicos apresentados a seguir:

◇ Conjunto de sinapses - cada sinapse é caracterizada por uma ponderação (ou peso), que quando positiva significa que a sinapse é excitativa e quando negativa é inibitória. Um sinal x_j na entrada da sinapse j conectada ao neurônio k é multiplicada por um peso sináptico w_{jk}.

◇ Somatório - para somar os sinais de entrada, ponderados pelas respectivas sinapses.

◇ Função de ativação - limita a amplitude da saída do neurônio.

A figura a seguir apresenta o modelo de um neurônio artificial.



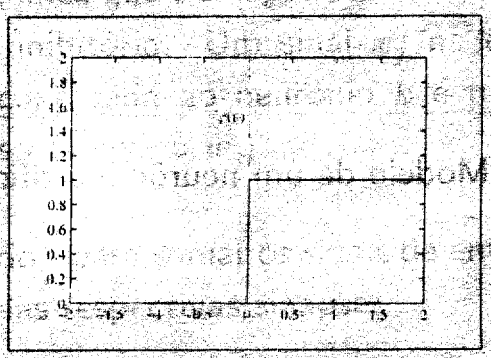
Modelo de um neurônio artificial

FUNÇÃO DE ATIVAÇÃO

A função de ativação pode ser de vários tipos:

1- Função indicadora (degrau)

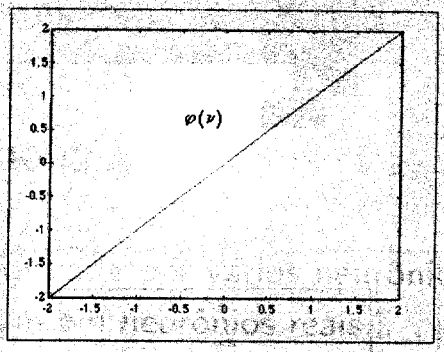
$$\varphi(\nu) = \left\{ \begin{array}{l} 1, \ \nu > b \\ 0, \ \nu < b \end{array} \right\}$$



Função indicadora

2- Função identidade (rampa)

$$\varphi(\nu) = \nu$$



Função identidade

3 - Função logística (sigmóide)

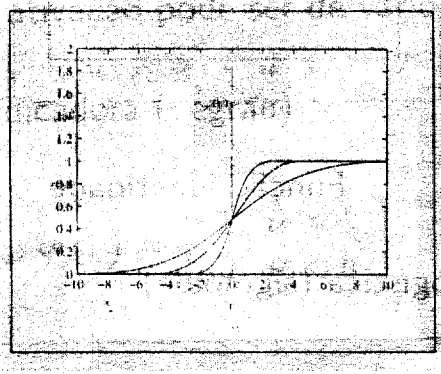
$$\varphi(\nu) = \frac{1}{1+\exp(-a\nu)},\ \nu \epsilon (0,1)$$

4 - Função tangente hiperbólica (sigmóide)

$$\varphi(\nu) = \tanh\left(\frac{\nu}{2}\right) = \frac{1-\exp(-\nu)}{1+\exp(-\nu)},\ \nu \epsilon (-1,1)$$

5 - Função normal

$$\varphi(\nu) = \Phi(\nu)$$



Função sigmóide

REDES NEURAIS ARTIFICIAIS

• A *RNA* é formada por vários neurônios artificiais que se inspiram em neurônios reais.

• Cada neurônio artificial é constituído por uma ou mais entradas e uma saída. Estas entradas podem ou não ser saídas de outros neurônios e a saída pode ser entrada para outros neurônios. As entradas são multiplicadas por pesos e somadas com uma constante; este total então passa pela função de ativação. Esta função tem o objetivo de ativar ou inibir o próximo neurônio.

Matematicamente, descrevemos o k-ésimo neurônio da seguinte forma

$$u_k = \sum_{j=1}^{p} w_{kj} x_j \text{ e } y_k = \varphi(u_k - w_{ko})$$

onde $x_o, x_1, ..., x_p$ são as entradas; $w_{k1}, ..., w_{kp}$ são os pesos sinápticos; u_k é saída da combinação linear; w_{ko} é o viés ; $\varphi(\bullet)$ é a função de ativação e; y_k é a saída do neurônio.

Observando a figura e as equações anteriores notamos que a representação de uma *RNA* constituída de um neurônio tem uma semelhança com o modelo de regressão não-linear múltipla. As variáveis independentes são as entradas do neurônio, a saída é a variável dependente y_k, w_{ko} é o intercepto, w_{kp} são os coeficientes de regressão e $\varphi(\bullet)$ é a função não-linear.

A primeira camada de uma *RNA*, chamada de *camada de entrada*, é contituída pelas entradas x_p e a última camada é a *camada de saída*. As camadas intermediárias são denominadas de *camadas escondidas*.

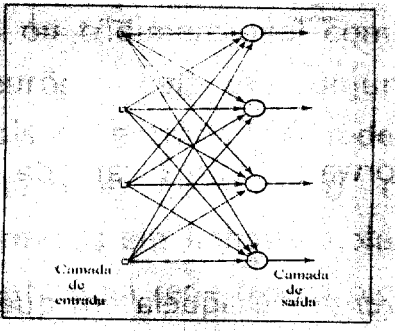
O número de camadas e a quantidade de neurônios em cada uma delas é determinada pela natureza do problema.

Um vetor de valores apresentado uma vez para todas as unidades de entrada de uma *RNA* é chamado de *caso, exemplo, padrão, amostra*, etc.

TIPO DE ARQUITETURA

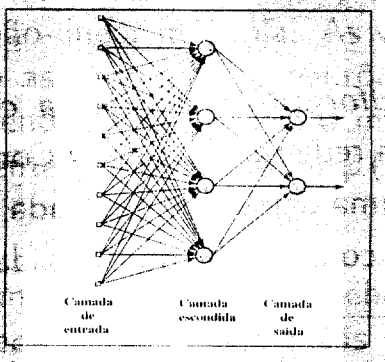
A arquitetura da rede neural se refere à organização dos neurônios e os tipos de conexões permitidas. A estrutura de uma RNA pode ser de vários tipos:

1. *Rede "feedforward " ou direta com duas camadas* - é a rede mais simples. Consiste de uma rede com uma camada de entrada (dados), que se conecta com a camada de saída.



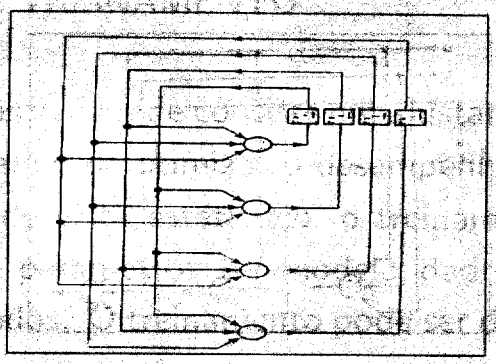
Rede "feedforward " com duas camadas

2 - *Rede "feedforward " ou direta* - é a rede em que os neurônios somente se conectam com os neurônios de camadas posteriores. Este tipo é o mais encontrado em aplicações estatísticas. A figura a seguir apresenta uma rede "*feedforward* " com uma camada escondida.



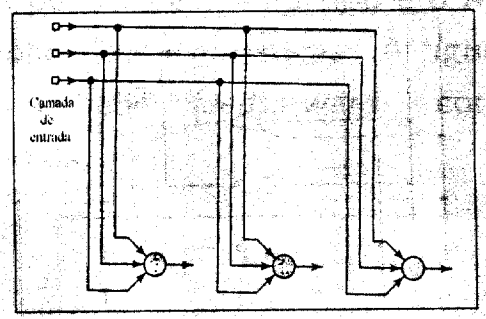
Rede "feedforward "com 1 camada escondida

3 - *Rede recorrente* - é aquela em que pelo menos um neurônio pode se conectar com neurônios de camadas anteriores. A figura abaixo apresenta uma rede recorrente sem camada escondida.

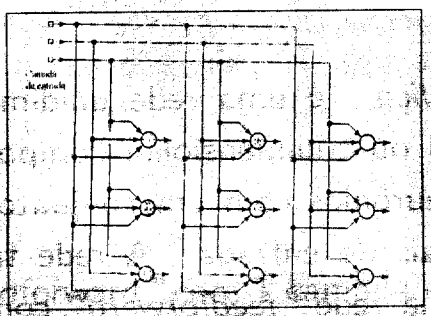


Rede recorrente

4 - *Rede treliça* - é uma rede unidimensional ou bidimensional ou tridimensional composta por um arranjo de neurônios com um conjunto correspondente de sinais de entrada. A rede treliça é basicamente uma rede "*feedforward*" com neurônios de saída arrumados em linhas e colunas. A figura abaixo apresenta uma rede treliça bidimensional.



: Rede Treliça unidimensional com 3 neurônios



Rede Treliça bidimensional com 3 x 3 neurônios.

TIPOS DE TREINAMENTO

O treinamento da rede consiste em ajustar as sinapses com o objetivo de otimizar o desempenho da rede. Do ponto de vista estatístico, o treinamento seria estimar os parâmetros do modelo dado um conjunto de dados. O treinamento pode ser da seguinte forma:

a) *Supervisionado* - é aquele em que para cada vetor de entrada se conhece a saída, podendo-se então modificar os pesos sinápticos da rede de forma ordenada a fim de atingir o objetivo desejado.

b) *Não-supervisionado* - o não-supervisionado difere do anterior pelo fato de não ter a saída para cada vetor de entrada. O treinamento não-supervisionado se iguala aos métodos estatísticos de análise de conglomerados e de componentes principais.

Para se estimar os pesos existe a necessidade de otimizar o desempenho sob uma amostra de treinamento minimizando uma função erro. Várias de aproximações são usadas para minimizar a função erro.

Uma RNA é caracterizada principalmente por três fatores:

1- *A arquitetura da rede* - organização dos neurônios e os tipos de conexões permitidas entre eles.

2 - *O tipo de treinamento* - método utilizado para estimar os parâmetros da rede para desempenhar uma determinada tarefa.

3 - *A função de ativação* - função que excita ou inibe o neurônio.

Quanto ao terceiro fator citado acima, a função de ativação mais usada para neurônios na camada

escondida é a função logística pois não é apropriado usar funções lineares porque a entrada para o neurônio na camada escondida é sempre uma combinação linear de saídas anteriores. Entretanto, para os neurônios da camada de saída, a função de ativação dependerá da natureza do problema. Por exemplo, para a predição de séries temporais , a variável de saída é contínua e a função identidade é apropriada e, para o problema de classificação, o resultado é 0 ou 1 e uma função limiar linear ou uma função logística deve ser preferida.

APLICAÇÕES DA RNA

- Reconhecimento de fala

- Localização de fontes de radar

- Otimização de processos químicos

- Reconhecimento automático de caracteres escritos a mão

- Reconhecimento de padrões e diagnósticos

- Classificação

- Detecção de sinais

- Controle de trajetória

NEURÔNIO DE MCCULLOCH-PITTS

O modelo de McCulloch-Pitts de neurônio é simplesmente uma unidade de limiar binária. O neurônio recebe a soma ponderada de entradas proveniente de unidades conectadas e tem como saída o valor um se sua soma for maior do que determinado limiar ou zero se sua soma for menor do que este limiar. Matematicamente, podemos representar este modelo como

$$u_k = \varphi\left(\sum_{j=1}^{p} w_{kj}x_j - \mu_k\right)$$

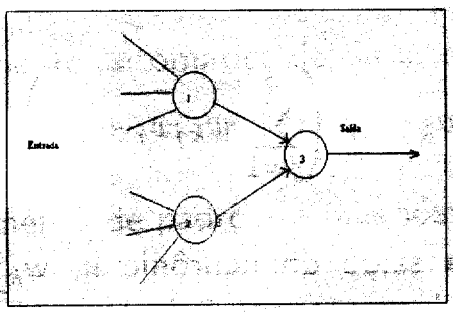
onde u_k é a saída do neurônio k, w_{kj} é o peso do neurônio j para o neurônio k, x_j é a saída do neurônio j, μ_k é o limiar para o neurônio k e, φ é a função de ativação, definida como

$$\varphi(entrada) = \begin{cases} 1, \text{ se a entrada } \geq 0 \\ 0, \text{ c.c} \end{cases}$$

PERCEPTRON DE ROSENBLATT

Em 1958, Rosenblatt criou um modelo com vários neurônios de McCulloch-Pitts, que chamou de *Perceptron*.

A figura abaixo apresenta uma rede *Perceptron* com 3 neurônios formando duas camadas.



Perceptron com 3 neurônios

Como já foi visto, a camada de entrada recebe os dados de fora e a camada de saída envia informações do neurônio para fora. O *Perceptron* de Rosenblatt utiliza as entradas e saídas binárias (-1 ou 1). O

Perceptron é equivalente a um discriminante linear, recordando que um discriminante linear é simplesmente uma função escore ponderada da seguinte forma,

$$\sum_j w_j x_j - w_o$$

Os pesos, w_i, podem assumir valores reais, as características (ou variáveis) são descritas como entradas, x_i, e w_o é uma constante. Geometricamente falando, em duas dimensões, a constante w_o é o intercepto.

Em um *Perceptron* de uma única saída, tem-se um número fixo de entradas, x_i, correspondendo ao número de variáveis ou características para uma aplicação específica. As entradas são multiplicadas por pesos e somadas com uma constante. O *Perceptron* produz uma saída indicando se pertence a classe 1 ou a classe 0. Quando a soma é maior do que 0, a saída é 1, e a classe selecionada é 1. Caso contrário,

saída é 1, e a classe selecionada é 1. Caso contrário, a saída é 0, e a classe 0 é selecionada, i.e.,

$$O = \begin{cases} 1 & \text{se } \sum w_i x_i + w_o > 0 \\ 0 & c.c \end{cases}$$

A constante é referida como limiar ou viés, porque indica que a soma dos produtos ponderados deve exceder -w_o. Geometricamente, o *Perceptron* seleciona a classe 1 para pontos acima da linha e classe 0 para pontos abaixo da linha no caso bimensional, caso com duas entradas.

EXEMPLO: Considere somente duas entradas no *Perceptron*. Suponha x_i=2, x_2=1, w_1=.5, w_2=.3 e w_o =-1. A saída deste perceptron é 1, porque (2)(.5)+(1)(.3)-1=.3 é maior do que 0.

No *Perceptron*, seus pesos constantes e as variáveis são as entradas. Como o objetivo é obter (ou estimar) os pesos ótimos, da mesma maneira que outros discriminantes lineares, o próximo passo é treinar o *Perceptron*. O *Perceptron* é treinado com amostras

e usa um procedimento de aprendizado sequencial para determinar os pesos. Este treinamento consiste em apresentar sequencialmente as amostras e para cada saída errada, os pesos são ajustados para corrigir o erro. Se a saída está correta os pesos não são ajustados.Como nenhuma suposição é feita sobre a população, este procedimento de treinamento é considerado não-paramétrico.

A equação abaixo descreve a forma geral de um procedimento iterativo que ajusta os pesos. A cada amostra apresentada ao *Perceptron*, um novo peso é calculado adicionando uma correção ao peso anterior. Formalmente, descrevemos o peso atual como $w_i(t)$, o peso no tempo t. O novo peso é $w_i(t+1)$, que será o peso atual no tempo t+1. O novo peso $w_i(t+1)$ é calculado por um fator de ajustamento, $\Delta w_i(t)$, para o peso atual, $w_i(t)$. O limiar, $\theta(t)$, é também revisado.

$$w_i(t+1) = w_i(t) + \Delta w_i(t)$$

$$\theta(t+1) = \theta(t) + \Delta\theta(t)$$

De acordo com o que foi visto acima o treinamento do *perceptron* é simplesmente um procedimento de correção de erro já que o ajuste só é feito quando ocorre um erro na saída. O objetivo do treinamento é encontrar a correção de cada peso w_i, $\Delta w_i(t)$, . O treinamento do *perceptron* é dado pela equação abaixo, onde T é a resposta correta ou verdadeira e O é a saída do perceptron.

$$\Delta w_i\,(t) = (T - O)\,x_i$$

$$\Delta\theta\,(t) = (T - O)$$

Para cada caso que é apresentado ao *perceptron* e cuja saída esteja correta nenhuma correção é feita nos pesos. Entretanto, quando a saída está errada, cada peso é ajustado subtraindo o valor correspondente no padrão de entrada, e 1 é subtraído ou adicionado ao limiar. Quando a resposta real é 1 e o perceptron diz 0, cada peso é ajustado adicionando

o valor correspondente no padrão de entrada e 1 é adicionado ao limiar caso contrário, 1 é subtraído do limiar.

EXEMPLO: No exemplo anterior, a saída do perceptron era 1 porque a soma de todos os pesos e o limiar era .3. Se a saída correta fosse 0, os pesos teriam sido ajustados como

$$w_1(t+1) = .5 + (0-1)\,2 = -1.5$$

$$w_2(t+1) = .3 + (0-1)\,1 = -.7$$

$$\theta(t+1) = -1 + (0-1) = -2$$

Os valores iniciais dos pesos são geralmente números aleatórios, entre 0 e 1. A apresentação sequencial das amostras continua indefinidamente até que algum critério de parada esteja satisfeito. Por exemplo, se 100 casos são apresentados e se não houver nenhum erro o treinamento estará completo. Entretanto, se houver algum erro, todos os 100 casos são apresentados de novo para o *perceptron*. Cada ciclo de apresentação é chamado de *época*.

O *teorema de convergência do perceptron* diz que quando duas classes são linearmente separáveis, o treinamento garante a convergência mas não com que velocidade. Assim, se existe uma discriminante linear que pode separar as classes sem cometer um erro, o procedimento de treinamento encontrará a linha ou o plano separador, mas não se sabe quanto tempo levará para encontrá-la.

Para tornar o aprendizado e a convergência mais rápida algumas modificações podem ser feitas. Como por exemplo:

• Normalizar os dados: todos os valores das variáveis de entrada devem estar dentro do intervalo 0 e 1.

• Introduzir uma taxa de aprendizado, $ltaxa$, no procedimento de atualização do peso, passando agora para $(ltaxa)(w_i)$. O valor de $ltaxa$ é um número entre 0 e 1.

ADALINE E MADALINE DE WIDROW

Um outro modelo neural, utilizando como método de treinamento o método de *Mínimos Quadrados* (LMS - Least Mean Square), foi desenvolvido por Windrow e Hoff, em 1960. Este modelo foi chamado de *ADAI N* (ADAptive LINear Element). Uma generalização multidimensional deste modelo é chamada de *MADA IN* (Múltipla *ADA IN*). A principal diferença funcional entre o *perceptron* e o *ADA-IN* está na maneira com que a saída do sistema de aprendizado é calculada. O *perceptron* faz com que a saída da rede seja mapeada em 0 ou 1. Já o sistema de aprendizado *M S* usa a saída da rede sem qualquer mapeamento dos valores de saída de 0 ou 1. Assim, dada uma entrada, a saída do *MS* é simplesmente o produto das entradas e os pesos somados com o limiar.

EXEMPLO - No exemplo anterior, a saída do perceptron era 1 porque (2)(.5)+(1)(.3)-1=.3 era maior do que 0. No LMS a saída é o próprio .3.

Para um dado padrão de entrada, a saída, O, difere do valor real , T, por T-O. A medida do desempenho de um LMS para um conjunto de amostras é simplesmente a soma dos quadrados das diferenças sobre todos os padrões, i.e.,

$$D_{LMS} = \sum_{p} (T_p - O_p)^2$$

onde p é o padrão específico de entrada e D_{LMS}, é o erro.

EXEMPLO - Se as amostras consistem de dois padrões de entrada e as saídas e as verdadeiras respostas são dadas na tabela abaixo, então a distância das respostas verdadeiras,

D_{LMS} é (1-.8)2+(0-.5)2=.29.

Real	Saída
1	0,8
0	0,5

Saídas para o exemplo do LMS

O objetivo do treinamento LMS é minimizar a distância média (quadrática) entre a resposta verdadeira e a saída.

EXEMPLO - Considere a saída do treinamento *LMS* era 0.3. Se a saída correta fosse 0, os pesos seriam ajustados assim,

$$w_1(t+1) = 0.5 + (0 - 0.3)\,2 = -0.1$$

$$w_2(t+1) = 0.3 + (0 - 0.3)\,1 = 0$$

$$\theta(t+1) = -1 + (0 - 0.3) = -1.3$$

Tanto o *perceptron* quanto o *LMS* procuram um separador linear. Quando as classes são linearmente separáveis, o *perceptron* encontra a linha ou plano que não fornece erro. O mesmo não acontece para classes não-linearmente separáveis. Neste caso, o *perceptron* não apresenta um bom desempenho. Já o *MS*, consegue um bom desempenho tanto para classes linearmente separáveis quanto para as não linearmente separáveis.

A técnica utilizada para que o treinamento LMS convirja para a distância mínima é chamada de gradient descent. As desvantagens deste método são:

1. Pode oscilar e não convergir.

2. Pode convergir para a resposta errada.

REDES MULTICAMADAS

O *Perceptron* e o ADALINE vistos anteriormente consideravam a *RNA* somente com camadas de entrada e de saída. Nesta seção veremos as redes com várias camadas.

No sistema de treinamento LMS descrito anteriormente, não existia mais de uma camada escondida. Para o *Perceptron*, uma vez calculada a soma ponderada, a ativação da unidade de saída era determinada pela função indicadora, isto é , a saída era 0 ou 1. Esta função de ativação produz uma não-linearidade, enquanto a do sistema de treinamento *MS* é completamente linear. A soma ponderada é diretamente utilizada para ativar a unidade de saída; nenhum mapeamento adicional é feito pela função de ativação. Assim, para redes com várias camadas, uma função de ativação seria a do procedimento de treinamento LMS com uma função continuamente

diferenciável.. Este não é o caso para o limiar lógico. Uma alternativa é usar como função de ativação a função sigmoidal ou logística.

Para qualquer número de valor real, a saída da função logística é um número entre 0 e 1, que é exatamente o intervalo válido para a probabilidade. A equação abaixo mostra que a saída ou ativação O_j, de uma j-ésima unidade de saída ou escondida, é calculada aplicando uma função logística na entrada da rede, N_j, da j-ésima unidade. A entrada do j-ésimo neurônio é a soma da tendência do j-ésimo neurônio, j, e a soma ponderada das saídas de todas as unidades conectadas com o j-ésimo neurônio. A saída de um neurônio é seu valor real, i.e., para o neurônio x_j, $O_j = x_j$

$$N_j = \sum_i w_{ij} O_i + \theta_j$$

$$O_j = \frac{1}{1 + \exp\left(-N_j\right)}$$

Este procedimento equivale ao treinamento *MS* para saídas não-lineares logísticas e é conhecido como *backpropagation*. Da mesma forma que o *MS*, o treinamento é iterativo, com os pesos ajustados depois da apresentação de cada caso. O procedimento de treinamento envolve um ajustamento dos pesos que é proporcional ao produto da taxa de aprendizado, $ltaxa$, um erro derivado, $errdrv$, e a entrada. Como existem várias camadas, a entrada para a unidade j pode ser a saída de uma unidade da camada anterior, i, assim,

$$\begin{aligned} w_{ij}(t+1) &= w_{ij}(t) + \Delta w_{ij}(t) \\ \theta_j(t+1) &= \theta_j(t) + \Delta\theta_j(t) \end{aligned}$$

$$\begin{aligned} \Delta w_{ij}(t) &= (ltaxa)(errdv)_j O_i \\ \Delta\theta_j(t) &= (ltaxa)(errdv)_j \end{aligned}$$

Primeiro todos os erros atuais são propagados da primeira camada para a última camada, i.e., da camada de entrada para a saída. Então, os erros, errdrv, são propagados da última camada para a camada precedente até atingir a primeira camada escondida. Os erros são resumidos nas equações abaixos para as unidades de saída na última camada e para os neurônios escondidos. Os erros da unidade j são calculados somando o erro de todos os neurônios que estão conectados com a unidade j na camada mais próxima.

$$(errdrv)_j = O_j(1-O_j)(T_j - O_j)$$

$$(errdrv)_j = O_j(1-O_j)(T_j - O_j)(\sum_k (errdrv)_k w_{jk})$$

EXEMPLO - Considere o problema do OU Exclusivo com as seguintes entradas e os pesos dadas abaixo.

Entradas e pesos para o problema do XOR

x_1	x_2	w_{13}	w_{14}	w_{23}	w_{24}	w_{35}	w_{45}	θ_3	θ_4	θ_5
1	2	.1	-.2	.3	.4	.5	-.4	.2	.3	.4

Suponha uma RNA com 1 camada escondida com 2 neurônios, 1 camada de entrada com 4 classes e um neurônio na camada de saída.

Cálculo dos pesos para o problema do XOR

Neurônio	Entrada	Saída
3	.1+.3+.2=.6	1/(1+exp(-.6))=.65
4	-.2+.4-.3=-.1	1/(1+exp(.1))=.48
5	.5*.65-.4*.48+.4=.53	1/(1+exp(-.53))=.63

EXEMPLO - A tabela abaixo apresenta o cálculo pelo backpropagation do erro para o exemplo do problema do XOR. A tabela abaixo apresenta os pesos e a tendência , com taxa de aprendizado de 1.

Neurônio	Erro	Ajuste da tendência
5	.63*(1-.63)*(0-.63)=-.147	-.147
4	.48*(1-.48)*(-.147)*(-.4)=.015	.015
3	.65*(1-.65)*(-.147)*(-.4)=-.017	-.017

Peso	Valor
w_{45}	-.4+(-.147)*.48=-.47
w_{35}	.5+(-.147)*.65=.40
w_{24}	.4+(.015)*1=.42
w_{23}	.3+(-017)*1=.28
w_{14}	-.2+(.015)*1=.19
w_{13}	.1+(-.017)*1=.08
θ_5	.4+(-.147)=.25
θ_4	-.3+(.015)=-.29
θ_3	.2+(-.017)=.18

Considere a contribuição do erro quadrático do p-ésimo exemplo de treinamento como Ep e o erro total como E. Ep é definido como a diferença quadrática entre a saída do neurônio e o valor observado da variável de saída para o p-ésimo exemplo de treinamento. Backpropagation como é usualmente aplicado é ``quase" um algoritmo steepest descent para minimizar E; as etapas do descent steepest são tomadas depois de cada exemplo de treinamento p para minimizar a componente Ep melhor do que usar uma etapa única para minimizar E.

Como o treinamento LMS, a taxa de aprendizado, ltaxa, tem um papel crítico na aplicação prática do Backpropagation. Para um dada rede e uma taxa de aprendizado infinitesimal, os pesos que tem o erro mínimo podem ser encontrados, mas pode demorar muito. Apesar do excelente ajuste dos dados de treinamento obtido pelo Backpropagation, o mesmo não acontece com os resultados da aplicação aos dados de testes independentes. Uma maneira de contornar esta dificuldade é considerar outras variações do Backpropagation, tais como:

Revisão por época ou caso - No exemplo do XOR, os erros, (errdrv)j, foram calculados e os pesos corrigidos depois da apresentação de cada caso. Na derivação matemática do treinamento Backpropagation, os pesos são corrigidos depois de

cada época, i.e., depois que os erros forem calculados para cada caso na amostra. Isto significa que o erro, (errdrv)j, é tomado como a soma de todos os erros para todos os n casos como dado na equação anterior. Embora a revisão por época tenha um fundamento teórico forte, a revisão por caso (padrão) obtém melhores resultados e é usada com mais frequência.

Apresentação aleatória ou sequencial - A época é a unidade de treinamento fundamental e o comprimento do treinamento frequentemente é medido em termos de épocas. Durante cada época de treinamento com revisão por padrões, os casos podem ser apresentados em ordem sequencial ou aleatória.

Estado (chute) inicial aleatório - Os pesos e as tendências são inicializados por números aleatórios no intervalo entre -0.5 a 0.5. (As entradas são frequentemente normalizadas por números entre 0 e 1.)

Taxa de aprendizado e mínimo local - O treinamento Backpropagation com uma taxa de aprendizado muito pequena torna o progresso muito lento e uma taxa de aprendizado muito grande apesar de tornar o progresso muito rápido produz oscilações entre soluções relativamente pobres. Essas características são detectáveis somente após experimentações. As taxas de aprendizado mais usadas estão entre 0 e 1.

Uma maneira ajudar na velocidade de convergência e a evitar mínimos locais é o uso do momento. Com o momento, os pesos são corrigidos combinando parte da revisão do novo peso indicado, $\Delta w_{ij}(t+1)$ com parte da revisão do peso anterior, $\Delta w_{ij}(t)$, isto é ,

$$\Delta w_{ij}(t+1) = (ltaxa)(errdv)_j O_i + (mom)\Delta w_{ij}(t)$$
$$\Delta \theta_j(t+1) = (ltaxa)(errdv)_j + (mom)\Delta \theta_j(t)$$

Geralmente o valor usado para a taxa de aprendizado é 0,5 e para o momento é 0,9. A utilização do momento deve permitir uma grande taxa de aprendizado que assim, dará velocidade a convergência e evitará o mínimo local. Por outro lado, a taxa de aprendizado de 1 com nenhum momento será mais rápida quando não houver problema com o mínimo local nem com a convergência.

Condições de parada - A solução é encontrada quando a distância do erro se reduz a zero ou a um valor que tende a zero. Entretanto, para uma rede neural arbitrária, a distância mínima pode não ser zero e pode ser muito grande. Alguns procedimentos podem ser usados para determinar quando parar:

Limitar o número de épocas - O treinamento termina para um número de épocas previamente estipulado.

Critério do progresso medido - A distância do erro pode ser amostrada e ponderada sob um número fixo de intervalos de épocas, por exemplo todas as 500 épocas de treinamento. Se a distância do erro médio para as 500 épocas mais recentes não for melhor do que os 500 anteriores, podemos concluir que não houve progresso e então o treinamento terminará .

E estas aproximações podem ser combinadas.

MEMóRIAS ASSOCIATIVAS DE HOPFIELD

Nas memórias associativas, cada classe é representada por um exemplar. Quando um padrão é observado (geralmente uma versão parcial ou modificada de um exemplar é apresentada) a memória identificará o exemplar correto sem ruído. Este tipo de rede neural se baseia na capacidade do cérebro em armazenar uma biblioteca de padrões e ser capaz de associar cada um deles com um novo padrão observado.

A rede de Hopfield é uma memória associativa que implementa as idéias de Hebb sobre o treinamento. Nesta rede, cada vetor característica, x, é um vetor binário (±1) multivariado. O objetivo é associar o x com um dos m exemplares armazenado na memória. Os exemplares armazenados são como um conjunto de treinamento consistindo de uma representação para cada tipo de classe. As saídas são estados estáveis de um procedimento iterativo, embora que termine em tempo finito.

Para entender como a rede de Hopfield processa um padrão de entrada, x, considere $y^{(0)}=x$ e calcule

$$y_i^{(n+1)} = f_h\left(\sum w_{ij} y_j^{(n)}\right), \qquad i = 1,\ldots,p; \qquad n = 0,1,\ldots$$

onde a matriz de pesos $W=\{w_{ij}\}$ é definida em termos dos exemplares $\{z^{(1)},...,z^{(m)}\}$ por

$$W = p^{-1}\sum_{j=1}^{m} z(j)\left(z(j)\right)^T$$

mas $w_{ij}=0$, para todo i. A maneira na qual W é construída é chamada de aprendizado Hebbiano.

A convergência do algoritmo é analisada através da superfície de energia.

$$L(y) = -\frac{1}{2}y^T W y$$

Do ponto de vista da teoria de otimização esta superfície de energia é análoga a função objetivo.

Para os estatísticos, a rede de Hopfield precisa da substituição da função de indicação pela sigmóide não-linear para que se tenha a interpretação probabilística. Assim, se y_i, for atualizada, torna-se y_i', onde

$$y_i' = \begin{cases} +1, & com\ \ probabilidade\ \ \left[1+\exp\left\{-\sum_j w_{ij}y_j\right\}\right]^{-1} = f_s\left(w_i^T y\right. \\ -1, & com\ \ probabilidade\ \ 1-f_s\left(w_i^T y\right) \end{cases}$$

APROXIMAÇÕES E MÉTODOS COMPUTACIONAIS PARALELOS

A maioria dos métodos de análise de dados pode ser modelado como uma aplicação:

$$f: A \rightarrow B$$

onde A e B são conjuntos finitos.

Exemplos

1) B um conjunto de categorias discretas e A um conjunto de vetores m-dimensionais descritivos caracterizando n-observações.
   A aplicação f é uma conglomeração e a escolha de f é o domínio da análise de conglomerados;

2) Em técnicas de redução de dimensionalidade, B é um espaço contendo n pontos que tem dimensão menor do que A. Neste caso o problema é encontrar a aplicação f, sem conhecimento preciso de B e o termo "não supervisionado" é utilizado na tarefa. Análise de componentes principais cai nesta categoria;

3) Classificação supervisionada é utilizado quando deseja-se determinar f porém algum conhecimento preciso de B é disponível. Análise de discriminante vai nesta categoria.

Tais algoritmos contem as seguintes fases:

- Aprendizado: onde f é determinada usando o conjunto de treinamento $B' \subset B$;

- Teste: verificação de como f se comporta no conjunto teste $B'' \subset B$, B" destinto de B'

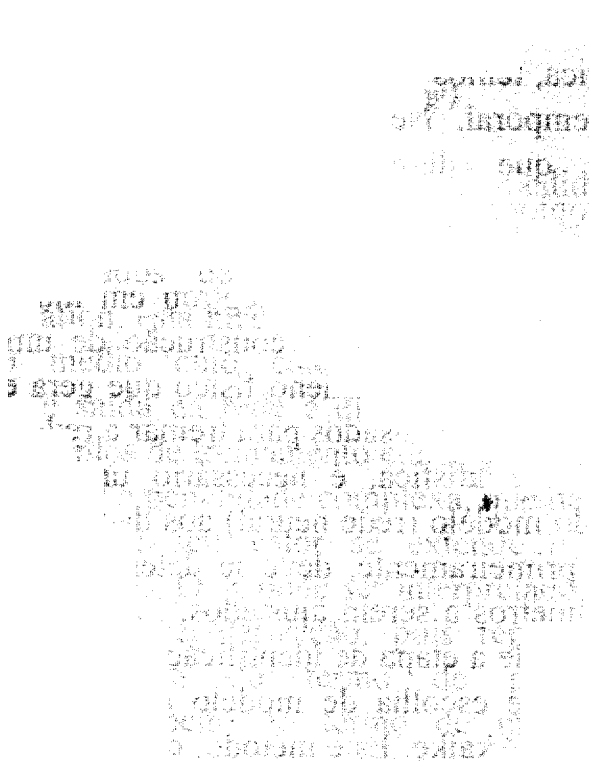
- Aplicação

A maioria das redes neurais (algoritmos para obter f) podem ser descrito como:

Determinar a f não linear onde $f: A \rightarrow B$ onde A é um conjunto de n vetores descritores m-dimensionais e B um conjunto de vetores p-dimensionais e onde alguns dos elementos de B são conhecidos.

A obtenção desta f é possível devido ao poderoso teorema de Andrei Kolmogorov sobre representação de funções contínuas e cuja adaptação a redes neurais é:

"Dada uma função continua f: $[0, 1]^n \rightarrow \Re^m$, $f(x) = y$, f pode ser implementada por uma rede neural com três camadas tendo um vetor de dimensão n na camada de entrada, $(2n + 1)$ neurônios na camada oculta e m neurônios na camada de saída".



ESCOLHA DA ESTRUTURA

É importante que as variáveis de entrada sejam normalizadas pelo fato da entrada dos neurônios na camada escondida ser uma combinação linear das varíaveis explicativas. é também importante ressaltar a existência de um dilema entre o viés e a variância. Um número grande de camadas e neurônios fornece um viés baixo ao mesmo tempo que apresenta uma variância grande. Na prática é comum tentar vários tipos de redes e usar a validação cruzada ou o desempenho de um conjunto de teste para escolher a rede mais simples que tem o melhor ajuste. Em estatística, temos o mesmo problema ao modelar uma série temporal. Neste caso, o modelo escolhido é aquele que descreve o sistema de maneira parcimoniosa para o objetivo escolhido.

O projeto de um Perceptron em multicamadas equivale, na realidade, a construção de um modelo não-linear de um fenômeno físico que gera exemplos de entrada e saída usados para treinar a rede neural. E, como na estatística, é necessário uma medida de ajuste do modelo (rede neural) aos dados observados. Então, primeiramente, deve-se determinar o número de parâmetros a serem ajustados. Em série temporal, isto equivale a etapa de identificação e existem vários métodos de escolha de modelo como por exemplo critério de Akaike. Este método, como os outros, tem

uma forma de composição comum, que é apresentada abaixo sendo que a diferença básica entre eles está no termo de penalidade da complexidade do modelo.

$$\begin{pmatrix}\text{Critério da}\\ \text{complexidade}\\ \text{do modelo}\end{pmatrix} = \begin{pmatrix}\text{função de log -}\\ \text{verossimilhança}\end{pmatrix} + \begin{pmatrix}\text{penalidade da}\\ \text{complexidade}\\ \text{do modelo}\end{pmatrix}$$

Seguindo o raciocínio acima, pode-se aplicar a mesma aproximação para o treinamento supervisionado. Sabe-se que o objetivo do aprendizado é encontrar um vetor de pesos que minimize o risco total

$$R(w) = \varepsilon_S(w) + \lambda\varepsilon_C(w)$$

O primeiro termo, $\varepsilon_S(w)$, é a medida do desempenho padrão, que depende tanto da rede quanto dos dados de entrada. No treinamento Backpropagation, esta medida é definida como o erro quadrático médio cujo cálculo se extende aos neurônios de saída da rede e que é executado para todos os exemplos de treinamento época por época. O segundo termo é a penalidade complexa, que depende somente da rede; seu valor se extende a todas conexões sinápticas. O λ pode ser interpretado como um parâmetro de regularização, pois representa a importância relativa do termo de penalidade complexa com respeito ao termo de medida do

desempenho. Quando λ é zero, o processo de treinamento Backpropagation não tem restrição, com a rede sendo determinada pelos exemplos de treinamento. Por outro lado, quando for muito grande a implicação é que a restrição imposta pela complexidade da penalidade seja por si só suficiente para especificar a rede, que é uma outra maneira de dizer que os exemplos de treinamento são irreais.

Terminologias usadas em modelos estatísticos e em redes neurais

Entrada da rede	Variáveis independentes, regressores
Saída da rede	Valores previstos
Valores de treinamento, alvos	Variáveis dependentes
Erros	Resíduos
Treinamento, aprendizagem, adaptação, organização própria	Estimação
Função erro, função custo	Critério de estimação
Padrões	Observações
Pesos	Estimativas dos parâmetros
Neurônios de alta ordem	Interações
Conexões funcionais	Transformações
Aprendizado supervisionado	Regressão e análise discriminante
Aprendizado não-supervisionado	Redução dos dados
Aprendizado competitivo	Análise de conglomerados

APLICAÇÕES EM ESTATÍSTICA

Important Statistical Models as special MLPs

1. Linear regression:

$$\begin{aligned} \psi(\boldsymbol{x}, \boldsymbol{\theta}) &= \boldsymbol{x}\boldsymbol{\theta} \\ \mu &= \psi(\boldsymbol{x}, \boldsymbol{\theta}) \end{aligned}$$

2. Linear discriminant analysis with two classes:

$$\begin{aligned} \psi(\boldsymbol{x}, \boldsymbol{\theta}) &= \boldsymbol{x}\boldsymbol{\theta} \\ \mu &= \begin{cases} 1 & \text{if } \psi(\boldsymbol{x}, \boldsymbol{\theta}) \geq 0 \\ 0 & \text{if } \psi(\boldsymbol{x}, \boldsymbol{\theta}) < 0 \end{cases} \end{aligned}$$

3. Multivariate regression:

$$\begin{aligned} \psi_k(\boldsymbol{x}, \boldsymbol{\theta}_k) &= \boldsymbol{x}\boldsymbol{\theta}_k, \qquad k = 1, \ldots, p \\ \mu_k &= \psi_k(\boldsymbol{x}, \boldsymbol{\theta}_k), \quad k = 1, \ldots, p \end{aligned}$$

4. Probit regression:

$$\begin{aligned} \eta &= \psi(\boldsymbol{x}, \boldsymbol{\gamma}) = \Phi(\boldsymbol{x}\boldsymbol{\gamma}) \\ \mu &= \begin{cases} 1 & \text{if } \eta \geq 0 \\ 0 & \text{if } \eta < 0 \end{cases} \end{aligned}$$

5. Logistic regression, logistic discriminant analysis:

$$\begin{aligned} \eta &= \psi(\boldsymbol{x}, \boldsymbol{\gamma}) = \frac{\exp(\boldsymbol{x}\boldsymbol{\gamma})}{1 + \exp(\boldsymbol{x}\boldsymbol{\gamma})} \\ \mu &= \begin{cases} 1 & \text{if } \eta \geq 0 \\ 0 & \text{if } \eta < 0 \end{cases} \end{aligned}$$

6. Generalized linear models:

$$\begin{aligned} \eta &= \boldsymbol{x}\gamma \\ \mu &= \psi(\eta) = g^{-1}(\eta) \end{aligned}$$

g is the inverse link function. Typical examples for a link function are:

$$\begin{aligned} \mu &= \psi(\eta) = \eta && \text{identic link function} \\ \mu &= \psi(\eta) = \frac{\exp(\eta)}{1+\exp(\eta)} && \text{logistic link function} \\ \mu &= \psi(\eta) = \exp(\eta) && \text{loglinear model} \end{aligned}$$

7. Generalized additive models with known smoothing function:

$$\begin{aligned} \xi_{jk} &= \psi_k(x_j)\,,\ j = 1,\ldots,r,\ k = 1,\ldots,K \\ \eta_j &= \sum_{k=1}^{K} \beta_{jk}\xi_{jk}\,,\ j = 1,\ldots,r \\ \mu &= \alpha + \sum_{j=1}^{r} \eta_j \end{aligned}$$

8. Projection pursuit models:

$$\begin{aligned} \zeta_l &= \boldsymbol{x}\gamma_l\,,\ l = 1,\ldots,L \\ \xi_{lk} &= \psi_k(\zeta_l)\,,\ k = 1,\ldots,K \\ \eta_l &= \sum_{k=1}^{K} \beta_{lk}\xi_{lk}, \quad l = 1,\ldots,L \\ \mu &= \alpha + \sum_{l=1}^{L} \eta_l \end{aligned}$$

9. LISREL models:

$$\begin{aligned} \eta &= B\eta + \Gamma x + \zeta \quad \text{structure model} \\ y &= \alpha + \Lambda\eta + \varepsilon \quad \text{factor model} \end{aligned}$$

These models may be written in the reduced form as:

$$\begin{aligned} \eta &= (I - B)^{-1}\Gamma x + (I - B)^{-1}\zeta \\ y &= \alpha + \Lambda(I - B)^{-1}\Gamma x + \Lambda(I - B)^{-1}\zeta + \varepsilon \end{aligned}$$

Thus, μ can be written as:

$$\mu = \alpha + \Lambda(I - B)^{-1}\Gamma x$$

Note, that only the mean structure $E(y|x)$ and not the covariance structu $V(y|x)$ is used to estimate the weights in artificial neural networks.

10. LISREL models with mixed dependent variables:

$$\begin{aligned} \eta &= B\eta + \Gamma x + \zeta \\ y^* &= \alpha + \Lambda\eta + \varepsilon \end{aligned}$$

The observed variables y_k are metric, dichotomous or ordinal and ar lated to y_k^* through a threshold model. These models can be written as

$$\begin{aligned} \eta &= (I - B)^{-1}\Gamma x + (I - B)^{-1}\zeta \\ y^* &= \alpha + \Lambda(I - B)^{-1}\Gamma x + +\Lambda(I - B)^{-1}\zeta + \varepsilon \\ \mu^* &= \alpha + \Lambda(I - B)^{-1}\Gamma x \\ \hat{y}_k &= g(\mu_k^*) \end{aligned}$$

$g(\mu_k^*)$ is one of the transfer functions given in section 1.2.

In practice, the following transfer functions are commonly used which allow generation of bounded or unbounded outputs:

$$g_k(e_k, \theta_k) = e_k\theta_k, \qquad \text{linear combination, } (-\infty, \infty)$$

$$g_k(e_k, \theta_k) = \Phi(e_k\theta_k), \qquad \text{normal distribution function, } (0, 1)$$

$$g_k(e_k, \theta_k) = \frac{\exp(e_k\theta_k)}{1 + \exp(e_k\theta_k)}, \qquad \text{logistic distribution function, } (0, 1)$$

$$g_k(e_k, \theta_k) = \tanh(e_k\theta_k), \qquad \text{hyperbolic tangent, } (0, 1$$

$$g_k(e_k, \theta_k) = \begin{cases} 1 & \text{if } e_k\theta_k > 0 \\ 0 & \text{if } e_k\theta_k \leq 0 \end{cases} \qquad \text{indicator function, } \{0, 1$$

$$g_k(e_k, \theta_k) = m \Longleftrightarrow \tau_{m-1} < e_k\theta_k \leq \tau_m, \quad m = 1, \ldots, M \qquad \text{threshold relation, } \{1, \ldots, M\}$$

Artificial Neural Networks	Statistics
network inputs	independent variables, regressors
network outputs	predicted values
training values, targets	dependent variables
errors	residuals
training, learning, adaptation, self-organisation	estimation
error function, cost function	estimation criterion
pattern	observations
weights	parameter estimates
higher order neurons	interactions
functional connections	transformations
supervised learning	regression and discriminant analysis
unsupervised learning, coding	data reduction
competitive learning	cluster analysis
generalization	interpolation and extrapolation

Tab. 1: Terminologies of ANN and statistics

REGRESSÃO

Regressão é usado para modelar relação entre variáveis. As covariáveis (variáveis independentes, regressores, estímulo), são denotados por x_i . Estas variáveis estão sob controle ou são observados. A variável resposta (dependente) é denotada por y. O objetivo da regressão é predizer ou classificar a resposta y a partir da covariável x_i . Outras vezes se deseja testar hipóteses sobre a relação funcional entre resposta e estímulo.

A forma geral do modelo de regressão é:

$$\eta = \sum_{i=0}^{I} \beta_i x_i$$

com

$$\eta = h(\mu) \quad e \quad E(Y) = \mu$$

Aqui h(.) é a função de ligação, β_i são os coeficientes, I é o número de covariáveis e β_0 o termo de interceptro.

Este modelo tem três componentes:

1. Um componente aleatório da variável resposta y com média μ e variância σ^2;

2. Um componente sistemático que relaciona o estímulo x ao preditor linear

$$\eta = \sum_{i=0}^{I} \beta_i x_i$$

3. uma função ligação que relaciona a média com o preditor linear

$$\eta = h(\mu)$$

O modelo linear generalizado reduz-se ao modelo de regressão linear quando a componente aleatória ~ $N(0, \sigma^2)$ e h(.) é a função identidade, neste caso:

$$y_j = \beta_0 + \sum_{i=1}^{I} \beta_i x_{ji} + \epsilon_j$$

Onde $\varepsilon_j \sim N(0, \sigma^2)$.

O objetivo do problema é obter os coeficientes β_i que minimizem

$$E = \sum_{j=1}^{n} \left(y_j - \sum_{i=0}^{I} \beta_i x_{ji} \right)$$

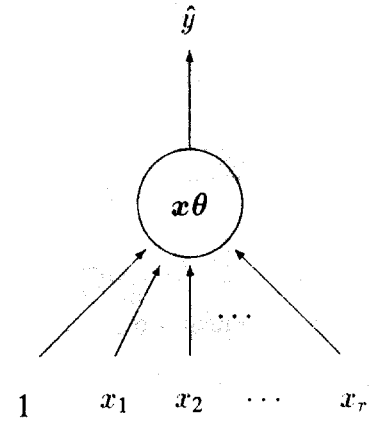
Baseado nas observações (x_j, y_j), $j = 1, \ldots, n$.

Este problema é equivalente a rede neural com uma camada da figura abaixo.

As covariáveis correspondem as entradas, a resposta y, as saídas e os coeficientes β's, aos pesos.

A função de ativação é a identidade. Os pesos são obtidos por um processo iterativo ao passo que uma expressão fechada é obtida na análise clássica de regressão.

Em geral qualquer modelo linear generalizado é equivalente a uma rede neural com uma camada. A função de ativação é escolhida para coincidir com o inverso da função de ligação $h = g^{-1}$ e a função objetivo é a deviance.



. 2: The ADALINE Network is equivalent to a univariate linear regression/discriminant analysis model.

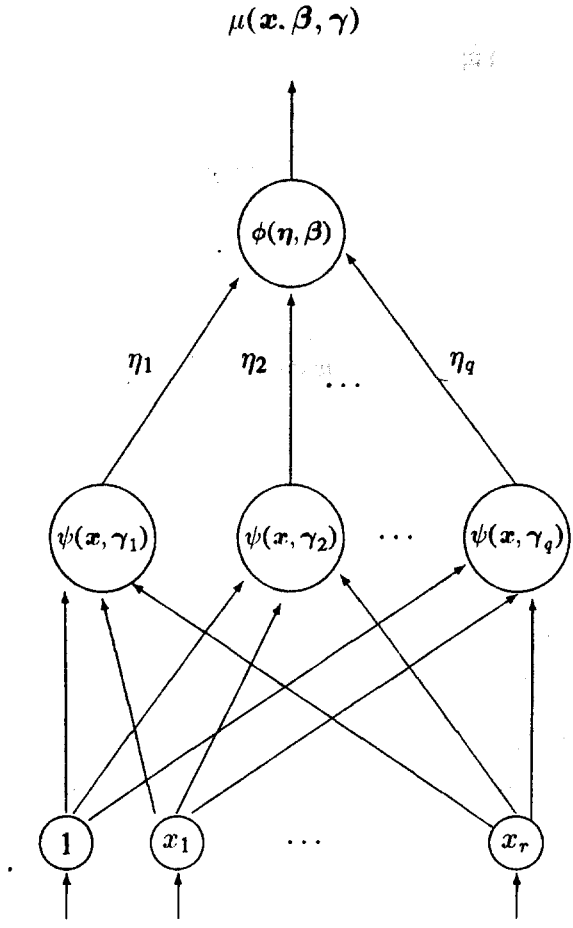


Fig. 3: A feedforward network with a single hidden layer.

EXEMPLO – REGRESSÃO

Church e Curram (1996) compararam previsões de despesa pessoal obtidas com rede neural e modelos econométricos (London Business School, National Institute of Economic and Social Research, Bank of England e uma equação de Goldman Sache).

Nenhum dos modelos foi capaz de explicar a queda no crescimento das despesas no fim dos anos 80 e começo dos anos 90.

Um resumo desta aplicação de redes neurais é o seguinte:

i) as redes neurais utilizaram exatamente as mesmas covariáveis e observações usadas em cada modelo econométrico. Estas redes produziram resultados semelhantes aos modelos econométricos;

ii) as redes neurais usaram 10 neurônios e uma camada escondida para todos os modelos;

iii) a variável dependente e as covariáveis foram re-escaladas para cair no intervalo 0,2 a 0,8 (ver Smith, 1993, p. 167);

iv) um exercício final utilizou uma rede neural com entradas de todas as variáveis de todos os modelos. Esta rede foi capaz de explicar a queda de crescimento, mas é uma rede com muitos parâmetros;

v) além de comparação das previsões também foi realizada uma análise de sensibilidade com respeito a cada variável. Para isto testou-se a rede com o valor médio dos dados e variou-se cada variável para verificar o grau em que cada variação afeta a previsão da variável dependente.

REGRESSÃO LOGÍSTICA

Consideremos uma situação onde observamos uma variável *resposta binária y e um vetor* $x = (x_1, \ldots, x_I)$ *de covariáveis para* cada um dos n indivíduos.

O modelo de regressão logística relaciona y a x supondo que:

$$P_i\{y=1/x\} = \Lambda\left(\beta_0 + \sum_{i=1}^{I} \beta_i x_{ji}\right)$$

Onde

$$\Lambda(u) = 1/\left(1+e^{-u}\right)$$

é a função logística.

Os coeficientes de regressão β são os log razão de chances ou e^{β} são as razões de chance.

O modelo logístico como um modelo de regressão não linear é um caso especial do modelo linar generalizado i.e

$$E(y/x) = \pi(x, \beta)$$

$$V(y/x) = \pi(x, \beta)(1 - \pi(x, \beta))$$

Usando a função de ligação logística:

$$\pi(x, \beta) = \Lambda\left(\beta_0 + \sum_{i=1}^{I} \beta_i \, x_{ji} \right) = \Lambda\left(\beta^T x\right)$$

Sendo $\beta = (\beta_0, \beta_1, \ldots, \beta_I)$ e adicionando $x_0 = 1$ ao vetor x. A estimação é baseada na maximização da função log-verossimilhança

$$L(\beta) = \sum_{j=1}^{N} \left[y_j \log \pi(x_j, \beta) + (1 - y_j) \log\left(1 - \pi(x_j, \beta)\right) \right]$$

Onde (x_j, y_j) são as observações para o indivíduo j.

Maximizar $L(\beta)$ é equivalente a minimizar a distância de Kulback-Leibler

$$\sum_{j=1}^{n} \left[y_j \log \frac{y_j}{\pi(x_j, \beta)} + (1 - y_j) \log \frac{1 - y_j}{1 - \pi(x_j, \beta)} \right]$$

Que pode ser escrita como

$$\sum_{j=1}^{n} - \log\left(1 - \left| y_j - \pi(x_j, \beta) \right|\right)$$

que torna a interpretação de distância entre y e $\pi(x, \beta)$ mais clara.

Para um indivíduo com vetor de covariáveis $x = (x_1, \ldots, x_I)$, a probabilidade $P(y = 1/x)$ é predita por

$$\hat{\pi} = \hat{\pi}(x, \hat{\beta}) = \Lambda(x\hat{\beta})$$

A rede neural com uma camada na figura abaixo, modelo de regressão também representa o modelo logístico.

Os pesos são obtidos por aprendizado usando a função de energia ou aprendizado:

$$E = \sum_{j=1}^{n} \left(y_j - \Lambda\left(\beta_0 + \sum_{i=1}^{I} \beta_i x_{ji} \right) \right)^2$$

Este processo é chamado retropropagação de mínimos quadrados.

Uma função de aprendizado alternativa é a distância de Kulback-Leibler e devido a relação com máxima verossimilhança é chamado retropropagação de máxima verossimilhança.

A extensão para o caso policotômico é obtido considerando $K > 1$ saídas.

A rede neural da figura abaixo é caracterizada por pesos β_{ik} ($i = 1, \ldots, K$) para as saidas y_k obtidos de

$$y_k = \Lambda\left(\beta_{0k} + \sum_{i=1}^{I} \beta_{ik}\, x_i \right)$$

Em termos estatísticos esta rede é equivalente ao modelo logístico multinomial ou policotômico definido por:

$$P(y = l/x) = \frac{\Lambda(x, \beta_l)}{\sum_{k=1}^{K} \Lambda(x, \beta_k)}$$

Os pesos podem ser interpretados como coeficientes de regressão e as estimativas de máximo verossimilhança são obtidas por retropropagação de máxima verossimilhança.

EXEMPLOS

Exemplo 1: Regressão Logística

Considere N = 39 respostas binárias denotando a presença (y = 1) ou a ausência (y = 0) de vasoconstrição na pele dos dedos depois da inspiração de um volume de ar à uma taxa de inspiração média R.

A Tabela apresenta os resultados da análise usual de dois modelos de regressão. O primeiro, considera somente uma covariância x_1 = log R, i.e., o modelo é o seguinte:

$$P(y=1/x)=A(\beta_0+\beta_1 x_1)$$

O segundo modelo considera o uso de uma segunda covariável, i.e.,

$$P(y=/x)=A(\beta_0+\beta_1+\beta_2 x_2)$$

Como x_1 = log R e x_2 = log V, respectivamente.

No primeiro caso, os resultados mostram um efeito marginalmente significante do logaritmo da taxa de inspiração na probabilidade para a presença da vasoconstrição na pele dos dedos.

Uma RN equivalente a este modelo seria a do perceptron coim dois neurônios de entrada ($x_0 \equiv 1$, x_1 = log R). Usando uma taxa de

aprendizado $\eta = 0{,}001$ para as últimas iterações ML – BP obtém-se $w_0 = -0{,}469$ e $w_1 = 1{,}335$ com uma distância de Kulback-Leibler de $E^* = 24{,}554$.

Tabela – Resultados das análises de regressão logística para os dados de vasoconstrição

Variável	Coeficiente de Regressão	Erro Padrão	Valor –P	Rede Neural
Intercepto	- 0.470	0.439	-	- 0.469
Log R	1.336	0.665	0.045	1.335
		(L = 24.543)		(E = 24.554)
Intercepto	- 2.924	1.288	-	- 2.938
Log R	4.631	1.789	0.010	- 4.651
Log V	5.221	1.858	0.005	5.240
		(L = 14.632)		(E = 14,73)

Para o caso do modelo com duas covariáveis, os resultados mostram a significante influência das duas covariáveis na probabilidade de vasoconstrição na pele dos dedos. Um perceptron com três neurônios de entrada ($x_0 = 1$, $x_1 = \log R$, $x_2 = \log V$) deveria dar resultados similares se ML – BP fosse usado. Isto quase foi obtido ($w_0 = 2{,}938$, $w_1 = 4{,}650$, $w_2 = 5{,}240$) embora o algoritmo backpropagation exija várias mudanças na taxa de aprendizado desde um valor inicial $\eta = 0{,}2$ até $\eta = 0{,}000001$ envolvendo

aproximadamente 20000 iterações para obter um valor da distância de Kullback-Leibler de E* = 14,73 comparável àquela de menos a log-verossimilhança (L = - 14,632).

Exemplo 2: Sonografia e Diagnóstico de Cancer da Mama

Medições de sonografia em N = 458 mulheres e características (x_1, ..., x_l) foram coletadas.

Verificou-se 325 (y = 0) tumores benignos e 133 (y = 1) tumores malignos.

Uma análise preliminar com um modelo logístico indicou três covariáveis como significantes, esta indicação mais problemas de colunearidade entre as 6 variáveis sugerem o uso de: idade, número de artérias no tumor (AT), número de artérias na mama controlateral (AC). Os resultados foram:

Logístico				
Variável	Coeficientes	Erro Padrão	Valor P	Rede Neural Pesos
Intercepto	- 8.178	0.924		W_0 = - 8.108
Idade	0.070	0.017	0.0001	W_1 = 0.069
log AT+1	5.187	0.575	0.0001	W_2 = 5.162
log AC+1	- 1.074	0.437	0.0014	W_3 = - 1.081
		L = 79.99		E = 80.003

Uma comparação com outras redes neurais é visto na tabela abaixo:

Exemplo 3: Sobrevida em operação cardíaca.

A variável dependente é uma variável binária, indicando se o paciente está vivo (y = 0), após 30 dias da operação de bypass arterial em caso contrário y = 1.

O objetivo é obter predições da probabilidade de sobrevivência dado os fatores de risco individuais As 12 variáveis independentes usadas incluem variáveis como idade, prioridade de operação, operações do coração prévias, etc.

Redes neurais e regresso logística foram usadas com 21.435 pacientes (2/3 das observações para treinamento) e 10.657 observações (1/3 para teste) para validação.

A performance preditiva dos dois modelos foi semelhante mas para calibração o modelo logístico foi muito melhor. (A medida usada foi o índice c e o teste Hosner e Lemeshow).

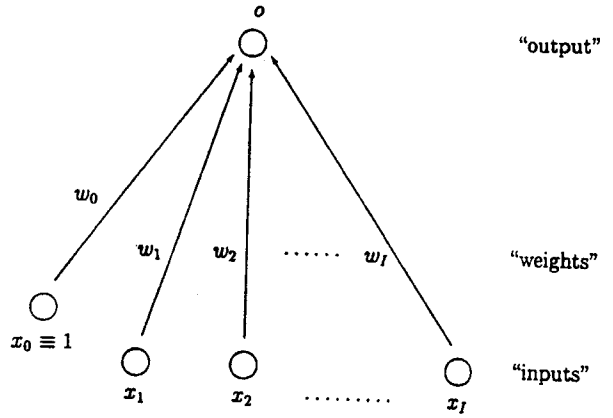


Fig. 1. The logistic perceptron.

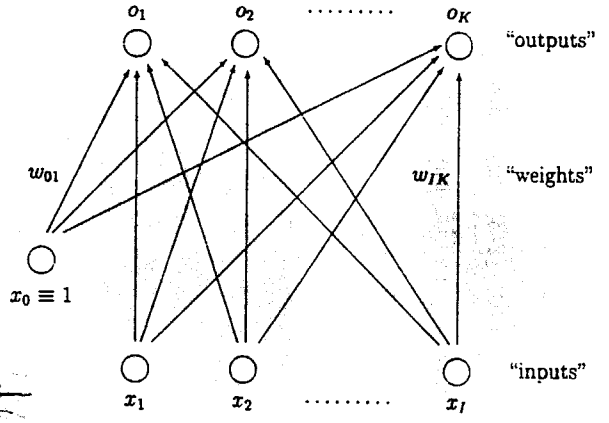


Fig. 2. The logistic perceptron with K output units.

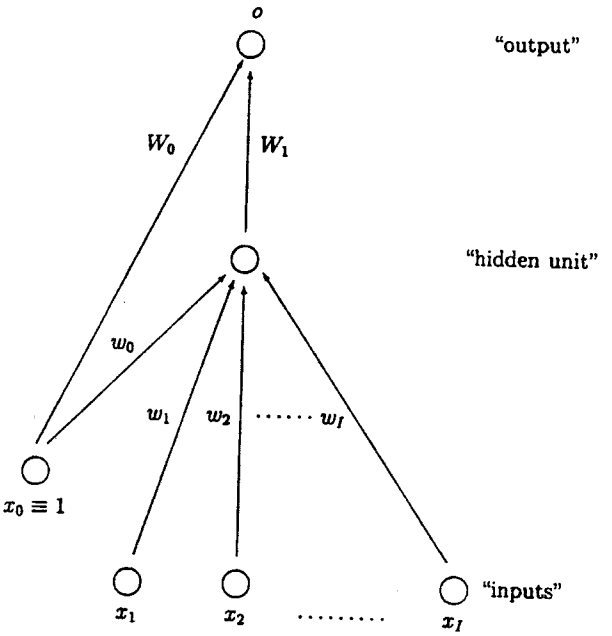


Fig. 3. Feed-forward neural network with one hidden unit.

$$o = f(x, w)$$
$$= \Lambda\left(W_0 + \sum_{j=1}^{J} W_j \cdot \Lambda\left(\sum_{i=0}^{I} w_{ij} x_i\right)\right).$$

Table 1
Results of the logistic regression analyses for the vasoconstriction data (model (5.1) upper part; model (5.2) lower part)

Variable	Estimated regression coefficient	Standard error	P-value
Intercept	−0.470	0.439	—
log R	1.336	0.665	0.045
		(L=−24.543)	
Intercept	−2.924	1.288	—
log R	4.631	1.789	0.010
log V	5.221	1.858	0.005
		(L=−14.632)	

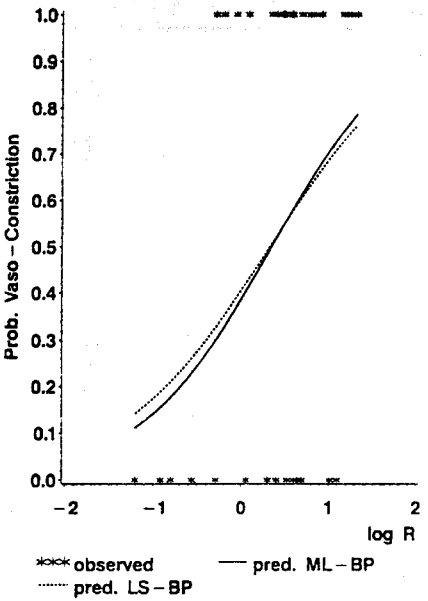


Fig. 4. Estimated response probabilities obtained by LS-BP (—) and ML-BP (- - - -) and observed responses (*) for the vasoconstriction data based on model (5.1).

Table 2
Results of a logistic regression analysis for the breast tumor data

Variable	Estimated regression coefficient	Standard error	*P*-value
Intercept	−8.179	0.924	—
Age (x_1)	0.070	0.017	0.0001
$\log AT + 1$ (x_2)	5.187	0.575	0.0001
$\log AC + 1$ (x_3)	−1.074	0.437	0.014
		($L = -79.996$)	

Table 3
Results of classification rules based on logistic regression, CART and feed-forward neural networks with J hidden units ($NN(J)$) for the breast tumor data

Method	Sensitivity (%)	Specificity (%)	Percentage of correct classifications
Logistic regression	95.5	90.2	91.7
CART	97.9	89.5	91.9
$NN(1)$	95.5	92.0	93.0
$NN(2)$	97.0	92.0	93.4
$NN(3)$	96.2	92.3	93.4
$NN(4)$	94.7	93.5	93.9
$NN(6)$	97.7	94.8	95.6
$NN(8)$	97.7	95.4	96.1
$NN(10)$	98.5	98.5	98.5
$NN(15)$	99.2	98.2	98.5
$NN(20)$	99.2	99.1	99.1
$NN(40)$	99.2	99.7	99.6

ANÁLISE DE SOBREVIVÊNCIA

O modelo de sobrevivência mais usado quando temos dados com covariáveis é o modelo de Cox com taxa de falha proporcional.

A taxa de falha é dada por

$$L(t, x_i) = h_0(t)\exp\{\beta x_i\}$$

O parâmetro β é estimado maximizando a verossimilhança parcial

$$L(\beta) = \prod_{i \in U} \frac{\exp(\beta x_i)}{\exp(\beta x_i)}$$

o produto é calculado nas falhas e R_i é a população sob riscos.

Redes neurais podem ser usadas em dados de sobrevivência nas seguintes formas:

1) Faragge e Simon, 1995, substituiram a função linear βx_i pela saída $f(x_i, \theta)$ da rede neural isto é

$$L_c(\theta) = \prod_{i \in U} \frac{\exp\left\{ \sum_{h=1}^{H} \alpha_h / \left[1 + \exp\left(- w_h' x_i\right)\right]\right\}}{\sum_{j \in R_i} \exp\left\{ \sum_{h=1}^{H} \alpha_h / \left[1 + \exp\left(- w_h' x_i\right)\right]\right\}}$$

e as estimativas são obtidas por máximo verossimilhança através de Newton-Raphson.

A rede correspondente é vista abaixo:

2) Liestol, Andersen e Andersen, 1994, usaram uma rede neural para o modelo de Cox com ℓ covariáveis na forma

Seja T a variável aleatória tempo de sobrevivência, e I_k o intervalo $t_{k-1} < t < t_k$, $k = 1, ..., K$ onde $0 < t_0 < t_1 < ... < t_K < \infty$.

O modelo pode ser especificado pelas probabilidades condicionais

$$P(T \in I_k / T > t_{k-1}, x) = \frac{1}{1 + \exp\left(- \beta_{0k} - \sum_{i=1}^{1} \beta_{ik} x_i \right)}$$

para k = 1, ..., K.

A rede neural correspondente é rede logística multinomial com k saídas.

A saída 0_k no k-ésimo neurônio de saída corresponde a probabilidade condicional de morrer no intervalo I_k.

Dados para o indivíduo n consiste do regressor x^n e o vetor $(y^n_1, ..., y^n_{Kn})$ onde y^n_k é o indicador do indivíduo n morrer em I_k e $k_n \leq K$ é o número de intervalos onde n é observado. Logo

$$y^n_1, ..., y^n_{K_n - 1} \text{ são todos } 0$$

$$\text{e} \quad y^n_{K_n} = 1$$

se n morre em I_{Kn}.

Sendo:

$$0_k = f(x, w) = \Lambda\left(\beta_{0k} + \sum_{i=1}^{I} \beta_{ik}\, x_i \right)$$

e a função a otimizar

$$E^*(w) = \sum_{h=1}^{N} \sum_{k=1}^{K_n} - \log\left(1 - \left| y_k^n - f\left(x^n, w\right) \right|\right)$$

e $w = (\beta_{01}, ..., \beta_{0k}, \beta_{11}, ..., \beta_{Ik})$ e para a hipótese de taxas proporcionais faz-se a restrição $\beta_{ij} = \beta_i$.

Outras implementações podem ser vistas em Biganzoli, Bosacchi, Mariani e Marubine, 1998.

Uma generalização imediata seria substituir a linearidade por não linearidade dos regressores adicionando uma camada oculta como na figura abaixo:

EXEMPLOS – ANÁLISE DE SOBREVIVÊNCIA

1) Faraggi e Simon, 1995 – Dados relacionados a 506 pacientes com câncer prostático nos estágios 3 e 4. As covariáveis são: estágio, idade, peso, tratamento (0, t, 1 ou 5 mg de DES e placebo).

   Os resultados são dados nas tabelas abaixo:

2) Lustol, Anderson e Anderson, 1994 – Dados de 205 pacientes com melonoma manigno dos quais 53 morreram, 8 covariáveis foram incluídas.

   Diversas redes foram estudadas e menos a verossimilhança (interpretado como erro de predição) são dados na tabela abaixo:

3) Biganzoli, Boracchi, Mariani e Marubine, 1998 – Aplicaram redes neurais nos conjuntos de dados:

- câncer de pescoço e cerébro (Efrom)
- câncer no pulmão (Kalbfleish e Prentices)

Os resltados são indicados abaixo:

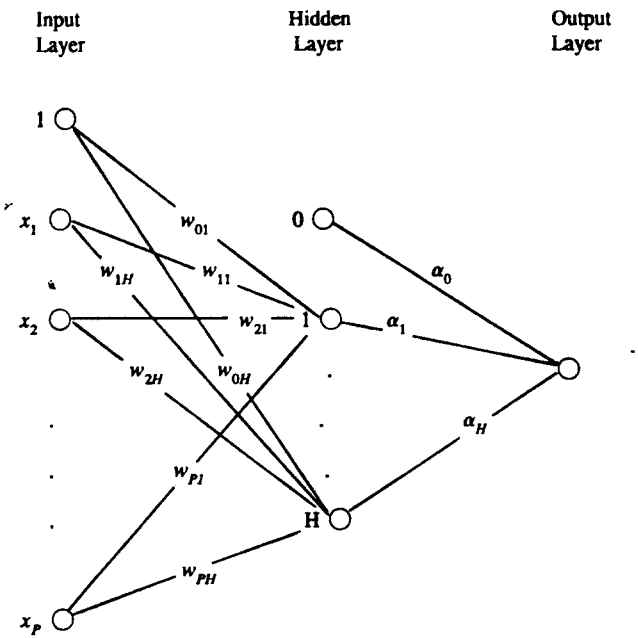


Figure 1. Single hidden layer neural network

he most commonly used model for censored data with covariates is Cox's proportional hazards PH) model.[13] The hazard function depends on time t and a vector of covariates x_i through

$$h(t,x_i) = h_0(t)\exp\{\beta x_i\}.$$

he vector of parameters β is estimated by maximizing the partial likelihood

$$L_c(\beta) = \prod_{i\in u} \frac{\exp(\beta x_i)}{\sum_{j\in R_i} \exp(\beta x_j)}$$

sing the Newton–Raphson method. The product in (3) is taken over the uncensored bservations. For more details see, for example, Miller.[18]

Consider replacing the linear functional βx_i in (2) by the output $g(x_i,\theta)$ of the network. The roportional hazards model becomes $h(t,x_i) = h_0(t)\exp[g(x_i,\theta)]$ and the function to be aximized becomes

$$L_c(\theta) = \prod_{i\in u} \frac{\exp\left\{\sum_{h=1}^{H} \alpha_h/(1+\exp(-w_h'x_i))\right\}}{\sum_{j\in R_i} \exp\left\{\sum_{h=1}^{H} \alpha_h/(1+\exp(-w_h'x_j))\right\}}$$

(a) First-order PH model (4 parameters);
(b) Second-order (interactions) PH model (10 parameters);
(c) Neural network model with two hidden nodes (12 parameters);
(d) Neural network model with three hidden nodes (18 parameters).

Table I. Summary statistics for the factors included in the models

	Complete data	Training set	Validation set
Sample size	475	238	237
Stage 3	47·5%	47·6%	47·4%
4	52·5%	52·4%	52·6%
Median age	73 years	73 years	73 years
Median Weight	98·0	97·0	99·0
Treatment: Low	49·9%	48·3%	51·5%
High	50·1%	51·7%	48·5%
Median survival	33 months	33 months	34 months
% censoring	28·8%	29·8%	27·8%

Table II. Log-likelihoods and c statistics for first-order, second-order and neural network proporational hazards models

Model	Number of parameter s	Training data		Test data	
		Log lik	c	log lik	c
First-order PH	4	− 815·3	0·608	− 831·0	0·607
Second-order PH	10	− 805·6	0·648	− 834·8	0·580
Neural network $H = 2$	12	− 801·2	0·646	− 834·5	0·600
Neural network $H = 3$	18	− 794·9	0·661	− 860·0	0·582

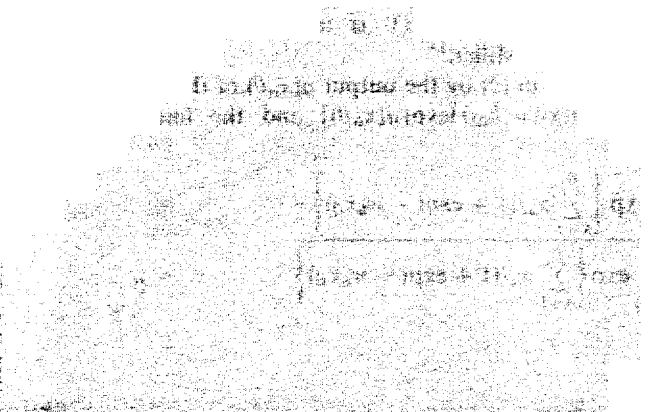
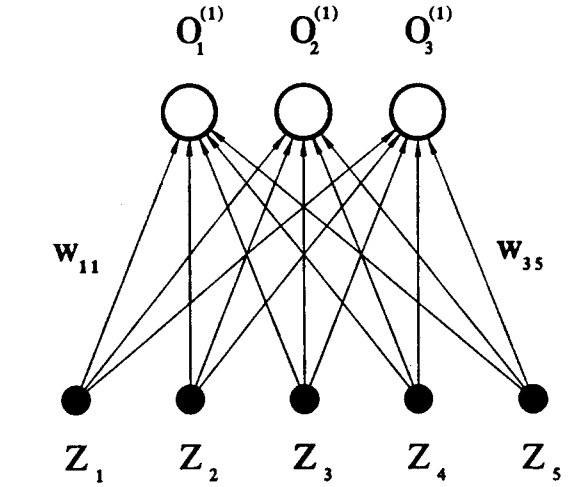


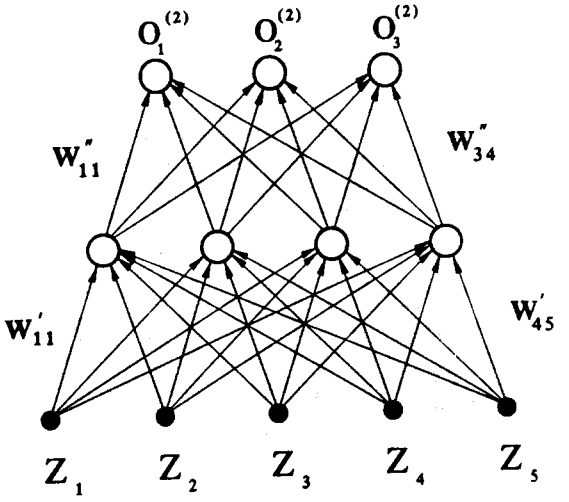
Table III. Estimation of the main effects and higher order interactions using 2^4 factorial design contrasts and the predictions obtained from the different models

Effects	PH 1st order	PH 2nd order	Neural network $H = 2$	Neural network $H = 3$
Stage	0·300	0·325	0·451	0·450
Rx*	− 0·130	− 0·248	− 0·198	− 0·260
Age	0·323	0·315	0·219	0·278
Weight	− 0·249	− 0·238	− 0·302	− 0·581
Stage × Rx	0	− 0·256	− 0·404	− 0·655
Stage × Age	0	− 0·213	− 0·330	− 0·415
Stage × Wt*	0	− 0·069	− 0·032	− 0·109
Rx × Age	0	0·293	0·513	0·484
Rx × Wt	0	− 0·195	− 0·025	0·051
Age × Wt	0	− 0·128	− 0·228	− 0·070
Stage × Rx × Age	0	0	0·360	0·475
Stage × Rx × Wt	0	0	0·026	0·345
Stage × Age × Wt	0	0	− 0·024	0·271
Rx × Age × Wt	0	0	0·006	− 0·363
Stage × Rx × Age × Wt	0	0	0·028	− 0·128

* Rx = Treatment
Wt = Weight



(a)



(b)

Figure 1. One layer (*a*) and two layer (*b*) feed-forward neural nets, showing the notation for nodes and weights: inpu
nodes (●), output (and hidden) nodes (○), covariates Z_j, connection weights w_{jk}, output values $O_k^{(s)}$

Table I. Cross-validation of models for survival with malignant melanoma: Column 1. Linear model; 2. Linear model with weight-decay; 3. Linear model with a penalty term for non-proportional hazards; 4. Non-linear model with proportional hazards; 5. Non-linear model with a penalty term for non-proportional hazards; 6. Non-linear model with proportional hazards in first and second interval and in third and fourth interval; 7. Non-linear model with non-proportional hazards

	1	2	3	4	5	6	7
'Prediction error'	172·6	170·7	168·6	181·3	167·0	168·0	170·2
Change		− 1·9	− 4·0	8·7	− 5·6	− 4·6	− 2·4

The main results for non-linear models with two hidden nodes were:

1. Proportional hazard models produced inferior predictions, decreasing the test log-likelihood of a two hidden node model by 8·7 (column 4) when using the standard weight decay, even more if no weight decay was used;
2. A gain in the test log-likelihood was obtained by using moderately non-proportional models. Adding a penalty term to the likelihood of a non-proportional model or assuming proportionality over the two first and last time intervals improved the test log-likelihood by similar amounts (5.6 in the former case (column 5) and 4.6 in the latter (column 6)). Using no restrictions on the weights except weight decay gave slightly inferior results (column 7, improvement 2.4).

In summary, for this small data set the improvements that could be obtained compared to the simple linear models were moderate. Most of the gain could be obtained by adding suitable penalty terms to the likelihood of a linear but non-proportional model (column 3).

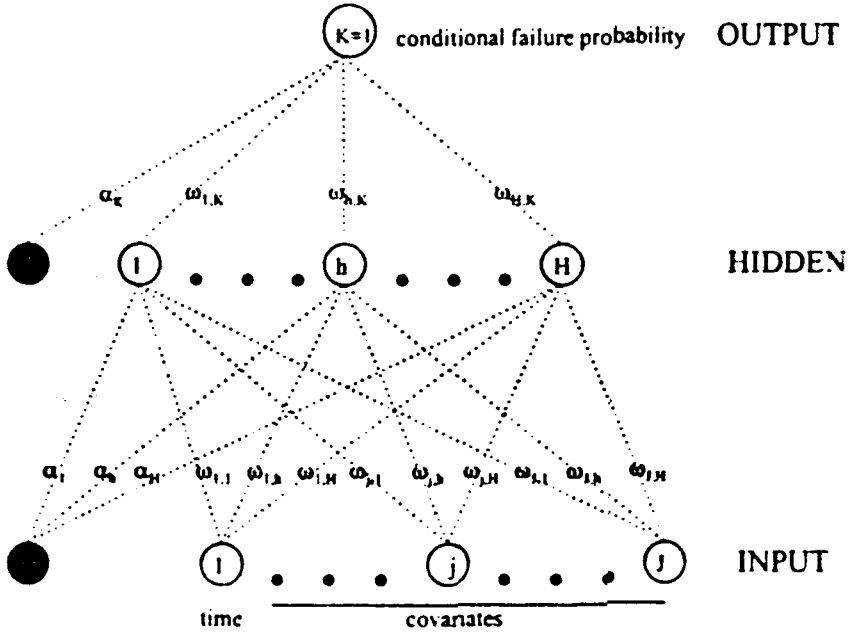


Figure 1. Feed forward neural network model for partial logistic regression (PLANN). The units (nodes) are represented by circles and the connections between units are represented by dashed lines. The input layer has J units, for time and the covariates, plus one bias unit (0). The hidden layer has H units plus the bias unit (0). A single output unit ($K = 1$) computes conditional failure probability. α_h and α_K are the weights for the connections of the bias unit with the hidden and output units. w_{jh} and w_{hK} are the weights for the connections between input and hidden units and hidden and output units, respectively.

Table I. Search for the best model for the head and neck trial data: values of NIC

Number of hidden nodes (H)	Penalty factor (λ)			
	0·025	0·05	0·075	0·1
2	99·31	100·92	138·46	104·56
3	100·61	99·96	102·36	103·64
4	99·88	103·04	102·27	102·04
5	100·91	100·40	99·73	103·97
6	98·50	102·56	99·36	100·11
7	98·69	99·40	99·63	99·78
8	98·93	100·46	99·38	99·47
9	98·85	99·63	98·71	99·74
10	98·37	102·67	98·71	105·88
11	98·94	99·69	98·09	100·41
12	99·36	98·51	97·81	99·05
13	99·07	98·82	98·29	98·63
14	99·07	98·84	97·83	98·94
15	99·52	99·56	98·44	99·51

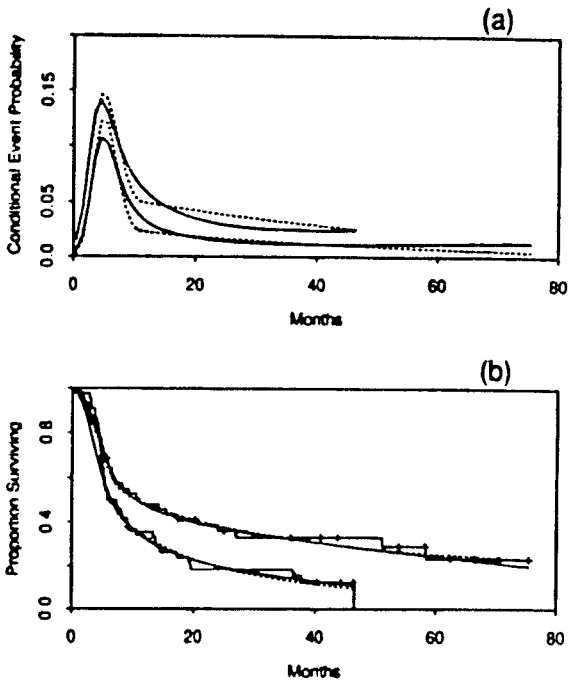


Figure 2. Head and neck cancer trial: (*a*) estimates of conditional failure probability obtained with the best PLANN configuration ($H = 12$, $\lambda = 0{\cdot}075$, solid line) and the cubic-linear spline proposed by Efron[13] (dashed line); (*b*) corresponding survival function and Kaplan–Meier estimates

Table II. Search for the best model for the VA lung cancer data: values of NIC

Number of hidden nodes (H)	Penalty factor (λ)			
	0·025	0·05	0·075	0·1
3	950·2	970·0	957·9	960·6
4	970·9	968·3	954·1	958·3
5	935·1	946·9	950·7	969·1
8	961·8	951·6	978·6	973·6

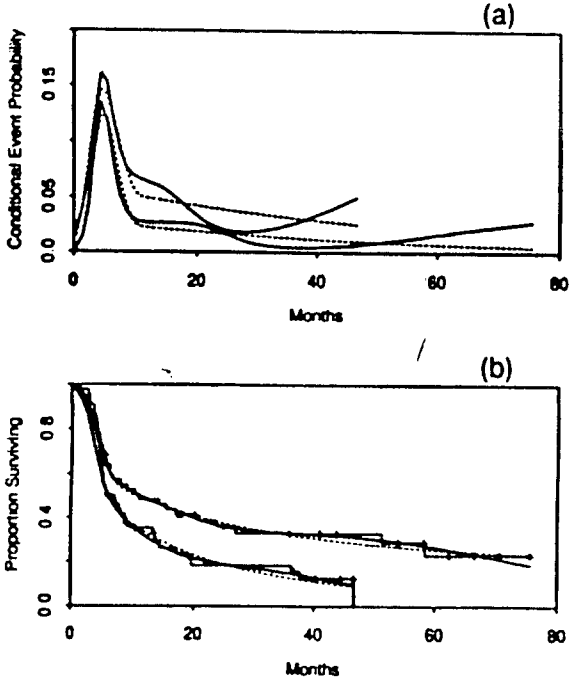


Figure 3. Head and neck cancer trial: (*a*) estimates of conditional failure probability obtained with a suboptimal PLANN model ($H = 12$, $\lambda = 0{\cdot}025$, solid line) and the cubic-linear spline proposed by Efron[13] (dashed line); (*b*) corresponding survival function and Kaplan–Meier estimates

ANÁLISE DE CONGLOMERADOS (CLUSTER)

No aprendizado não supervisionado, não existe como informar à rede se a sua saída está certa ou errada. Ela deve descobrir, por si só, padrões, correlações ou categorias nos dados de entrada e codificá-los na saída.

Esse tipo de procedimento só produz resultados aproveitável quando há redundância nos dados de entrada. Sem redundância seria impossível achar padrões ou similaridades nos dados, os quais se pareceriam com um ruído aleatório.

As técnicas que são empregadas nesse procedimento, ou criam padrões (classificação), ou medem a projeção em componentes principais dos dados de entrada. As técnicas de classificação ou aprendizado competitivo em que somente uma unidade de saída é ativada por vez (comumente chamadas de "winner-take-all") são equivalentes ao problema de análise de conglomerados.

Aprendizado Competitivo – Conglomerados

A meta das redes que empregam essa técnica é classificar os dados de entrada em categorias, de modo que entradas similares sejam agrupadas juntas e acionem a mesma unidade de saída. As classes devem ser encontradas pela própria rede a partir de correlações dos dados de entrada.

A arquitetura mais simples de redes desse tipo consiste em uma camada de saída, totalmente conectada às entradas por meio de conexões com pesos $w_i > 0$. A figura abaixo ilustra a arquitetura.

A unidade de saída vencedora (i*) desse procedimento é, normalmente, a que possui a "net" (h) maior. Assim:

$$h_i \geq h_i \quad \text{i.e.}$$

$$w_i\, z \geq w_i z, \quad \forall i$$

Se os pesos foram normalizados, de modo que $[w] = 1$, $\forall i$, então a equação acima é equivalente a:

$$|w_i - z| \leq |w_i - z|, \quad \forall i$$

indicando que o vencedor é o elemento de saída com o vetor de pesos normalizados w mais próximos ao vetor de entrada z.

Nota-se, portanto, que a rede do tipo "winner-take-all" implementa um classificador de padrões usando o critério das equações anteriores. Para atingir a meta de encontrar as classes nos dados de entrada é necessário, então, escolher os vetores de pesos w, adequadamente.

A regra de aprendizado competitivo padrão usa a equação

$$\Delta w_{rj} = \eta\left(z_i^{|r|} - w_{rj}\right)$$

para atualizar os pesos da unidade de saída vencedora em direção a entrada. Uma regra mais elaborada, onde os elementos de saída vizinhos ao vencedor também "aprendem", é a proposta no algoritmo de Kohonen

$$\Delta w_{ij} = \eta\Lambda(i, i^*)\left(z_j - w_{ij}\right), \quad \forall i, j$$

onde $\Lambda(i,i^*)$ é chamada função de vizinhança e vale 2 se $i = i^*$ e diminui com a distância $[r_i - r_{j^*}]$ entre as unidades de sa;ida i e i*. Dessa forma, elementos próximos ao vencedor e o próprio vencedor têm seus pesos alterados significativamente, enquanto aqueles distantes ($\Lambda(i,i^*)$ pequeno) são pouco afetados.

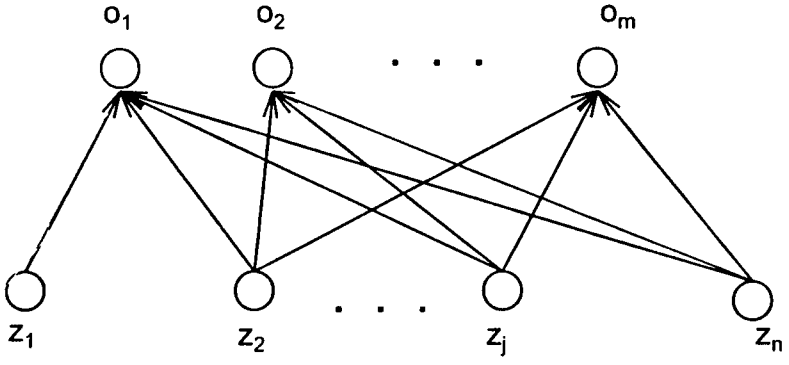


Figura 3-3 - Rede de Aprendizado Competitivo

EXEMPLOS - CLUSTER

Exemplo 1: (Mangiameh, Chen e West, 1996) - É uma comparação da performance de redes neurais e sete métodos hierárquicos de análise de conglomerados. Foram testados 252 conjuntos de dados com vários níveis de imperfeição que incluiam dispersão, observações aberrantes, variáveis irrelevantes e conglomerados não uniformes. Os resultados são apresentados abaixo:

Exemplo2: (Rogenthal, Engelhardt e Carvalho, 1998) - Parecem existir pelo menos três padrões funcionais em esquisofrenia. Esses padrões definem pelo menos tais grupos de sintomas ou dimensões de pacientes esquisofrenicos que são: aqueles que tem alucinações e pensamento confuso (dimensão psicótica), aqueles que perdem a motivação e auto-estima (dimensão negativa) e aqueles com pobreza de discurso e pensamento desorganizado (dimensão desorganizada).

Neste estudo 53 pacientes (fora de surto, não viciados, fisicamente capazes e com idade menor que 50 anos) responderam ao DSM-IV (Diagnostic and Statistical Manual for Mental Disorders) e submetidos a testes neuropsicológicos.

A rede neural (e um método de análise de conglomerados para comparação) indicaram 2 conglomerados importantes. Sendo que um deles baixo QI se mantem estável ao aumentar-se a busca para mais grupos.

Table 1
Data set design

Data set	Design factors	# of data sets
Base data	2, 3, 4, or 5 clusters; 4,6, or 8 variables; low, med, or high dispersion	36
Irrelevant variables	Basic data plus 1 or 2 irrelevant variables	72
Cluster density	Basic data plus 10% or 60% density	72
Outliers	Basic data plus 10% or 20% outliers	72

Table 2
Average distance between clusters

Number of clusters	Average distance (units)
2	5.03
3	5.31
4	5.33
5	5.78

Table 3
Basis for data set construction

Level of dispersion	Avg. cluster std. dev. (in units)	Range in std. dev.
High	7.72	3
Medium	3.72	6
Low	1.91	12

Table 4
SOM neural network cluster recovery rank

Dispersion	Low	Medium	High	Total	Percent
First	29	59	73	161	63.9%
Tied for first	24	5	1	30	11.9%
Second	19	9	1	29	11.5%
Not in top 2	12	11	9	32	12.7%

Table 5
Cluster definition results for base data percent correctly assigned

Dispersion	Low				
# Clusters	2	3	4	5	Average
Single linkage	67.7	84.4	75.1	80.3	76.9
Complete linkage	95.7	97.1	93.7	94.8	95.3
Average linkage	99.3	99.6	96.8	97.2	98.2
Centroid method	99.0	99.3	99.2	86.5	96.0
Ward's method	99.3	99.6	96.8	97.2	98.2
Two stage density	100.0	99.8	97.0	97.5	98.6
K th neighbor	51.0	67.3	67.3	86.7	68.1
SOM network	100.0	99.3	95.0	97.7	98.0
Dispersion	Medium				
# Clusters	2	3	4	5	Average
Single linkage	51.0	34.9	26.3	21.2	33.4
Complete linkage	85.7	72.4	61.5	59.9	69.9
Average linkage	51.7	44.9	35.3	41.6	43.4
Centroid linkage	51.3	34.4	28.5	32.7	36.7
Ward's method	95.3	96.0	79.2	74.4	86.2
Two stage density	64.3	73.8	38.5	43.6	55.1
K th neighbor	51.0	34.4	25.8	20.8	33.0
SOM network	100.0	100.0	93.0	90.9	96.0
Dispersion	High				
# Clusters	2	3	4	5	Average
Single linkage	51.0	34.2	26.0	21.3	33.1
Complete linkage	58.7	43.6	40.5	38.1	45.2
Average linkage	51.7	35.8	29.8	28.9	36.6
Centroid method	51.3	35.3	27.3	22.1	34.0
Ward's method	61.0	51.6	42.0	46.9	50.4
Two stage density	51.0	34.2	26.0	21.3	33.1
K th neighbor	51.0	34.4	26.2	21.2	33.2
SOM network	100.0	80.9	77.3	71.9	82.5

Table 6
Cluster definition results with outliers and medium dispersion level percent correctly assigned

Outliers	20%				
# Clusters	2	3	4	5	Average
Single linkage	50.0	44.4	25.0	20.0	34.9
Complete linkage	50.0	44.4	25.0	20.0	34.9
Average linkage	50.0	44.4	25.0	20.0	34.9
Centroid linkage	50.0	44.4	25.0	20.0	34.9
Ward's method	50.0	44.4	25.0	20.0	34.9
Two stage density	63.7	72.4	45.2	54.8	59.0
K th neighbor	50.0	44.4	25.0	20.0	34.9
SOM network	83.3	88.9	91.7	81.2	86.3
Outliers	40%				
# Clusters	2	3	4	5	Average
Single linkage	50.0	33.3	25.0	20.0	32.1
Complete linkage	50.0	33.3	25.0	20.0	32.1
Average linkage	50.0	33.3	25.0	20.0	32.1
Centroid method	50.0	33.3	25.0	20.0	32.1
Ward's method	50.0	33.3	25.0	20.0	32.1
Two stage density	63.7	73.6	45.2	43.1	56.4
K th neighbor	50.0	33.3	25.0	20.0	32.1
SOM network	83.7	88.7	91.5	70.0	83.5

Table 7
Cluster definition results with irrelevant variables and medium dispersion percent correctly assigned

Irrelevant	1 variable				
# Clusters	2	3	4	5	Average
Single linkage	51.0	34.7	26.0	21.1	33.2
Complete linkage	60.0	78.4	62.5	65.2	66.5
Average linkage	50.3	47.1	41.2	62.1	50.2
Centroid method	50.3	34.4	27.0	22.8	33.6
Ward's method	91.0	88.7	77.3	78.0	83.8
Two stage density	50.3	58.9	31.5	43.5	46.1
K th neighbor	50.3	34.2	26.2	21.1	33.0
SOM network	100.0	97.6	74.5	63.9	84.0
Irrelevant	2 variables				
# Clusters	2	3	4	5	Average
Single linkage	50.3	34.7	26.3	21.2	33.1
Complete linkage	65.0	60.2	53.7	59.6	59.6
Average linkage	50.0	54.9	49.3	54.7	52.2
Centroid method	50.3	34.7	28.0	21.5	33.6
Ward's method	80.7	90.0	70.2	76.3	79.3
Two stage density	50.3	52.9	25.8	39.9	42.3
K th neighbor	50.3	34.4	25.8	21.3	33.0
SOM network	100.0	90.7	69.0	57.2	79.2

Table 8
Cluster definition results with cluster density and medium dispersion percent correctly assigned

Density	10%				
# Clusters	2	3	4	5	Average
Single linkage	90.3	48.0	31.3	28.4	49.5
Complete linkage	60.7	73.0	68.5	67.5	67.4
Average linkage	90.3	75.1	52.8	75.2	73.4
Centroid method	90.3	46.2	43.0	32.8	53.1
Ward's method	66.7	80.7	82.2	82.4	78.0
Two stage density	89.7	93.1	64.5	62.8	77.5
K th neighbor	91.0	47.8	31.2	28.0	49.5
SOM network	89.0	83.1	90.8	90.5	88.4
Density	60%				
# Clusters	2	3	4	5	Average
Single linkage	60.3	61.1	60.8	61.1	60.8
Complete linkage	74.3	68.2	44.8	50.5	59.5
Average linkage	60.3	75.6	81.3	74.3	72.9
Centroid method	60.3	61.3	63.3	66.5	62.9
Ward's method	93.0	87.6	71.3	64.5	79.1
Two stage density	72.0	75.6	60.2	59.5	66.8
K th neighbor	60.3	61.1	57.8	60.9	60.0
SOM network	100.0	93.6	72.7	54.7	80.3

Table 9
Network weights and cluster centroids, high dispersion, 2 clusters, 6 variables

Cluster	#1					
Variable	1	2	3	4	5	6
$\alpha = 0.05$	0.7188	−0.0044	0.1630	−0.5761	0.1316	0.3215
$\alpha = 0.1$	0.7180	−0.0039	0.1644	−0.5754	0.1314	0.3210
$\alpha = 0.2$	0.7175	−0.0042	0.1655	−0.5753	0.1313	0.3209
$\alpha = 0.5$	0.7276	−0.0048	0.1543	−0.5759	0.1330	0.3202
Centroid	0.7184	−0.0043	0.1654	−0.5761	0.1328	0.3157
Cluster	#2					
variable	1	2	3	4	5	6
$\alpha = 0.05$	−0.7091	−0.3185	−0.0762	0.0286	−0.0733	−0.3389
$\alpha = 0.1$	−0.7093	−0.3197	−0.0766	0.0292	−0.0733	−0.3387
$\alpha = 0.2$	−0.7094	−0.3212	−0.0761	0.0295	−0.0735	−0.3389
$\alpha = 0.5$	−0.7096	−0.3220	−0.0740	0.0306	−0.0739	−0.3397
Centroid	−0.7006	−0.3166	−0.0736	0.0315	−0.0740	−0.3405

CLASSIFICAÇÃO

Em problemas de classificação podemos utilizar os seguintes procedimentos:

Se temos p variáveis descritoras de cada elemento e desejamos classificar o mesmo dentro de K classes:

1) Utilizamos o modelo logístico multinomial ou policotomico

$$p(y_i = 1/x) = \frac{\Lambda(x, \beta_i)}{\sum_{k=1}^{K} \Lambda(x, \beta_k)}$$

2) Podemos também utilizar redes múltiplas chamadas de "peritos locais" competindo para identificar diferentes aspectos do problema. Uma "rede de entrada" controla a competição e aloca diferentes regiões dos espaços dos dados às diferentes redes. A rede de entrada tem tantas saídas como o número de peritos locais e suas saídas são normalizadas para somar 1.

Por exemplo, se no caso anterior, temos

$$\text{logito } P(y_i = 1/x) = \beta x$$

aqui teremos

$$\text{logito } P(y_1 = 1/x) = g_1(z_1) + \ldots + g_p(z_p)$$

A figura abaixo ilustra esta rede.

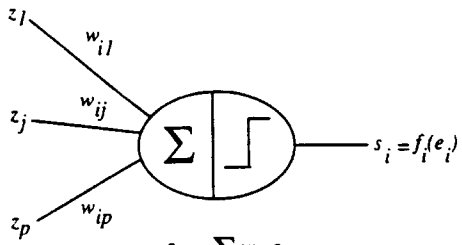
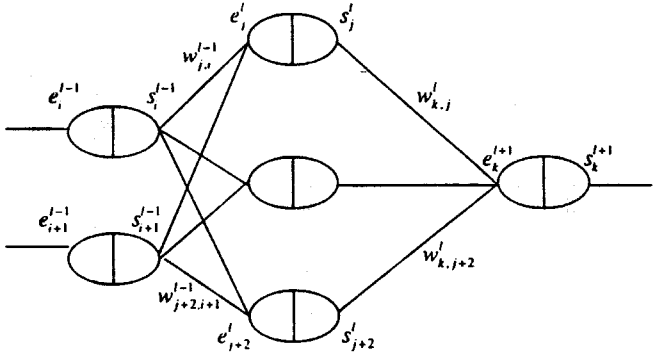


Figure 1. Formal neuron



where $e_i^{(l+1)} = \sum_{j=1}^{n_l} w_{ij}^{(l)} s_j^{(l)}$ and $s_i^{(l+1)} = f_j^{(l+1)}(e_j^{l+1})$

Figure 2. Multi-Layer Perceptron

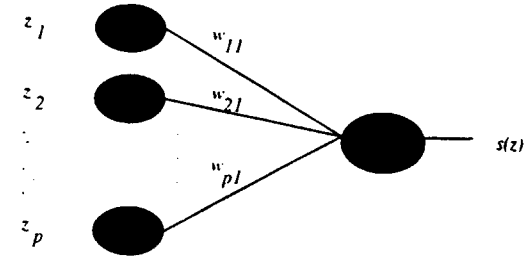


Figure 3. Neural Network.

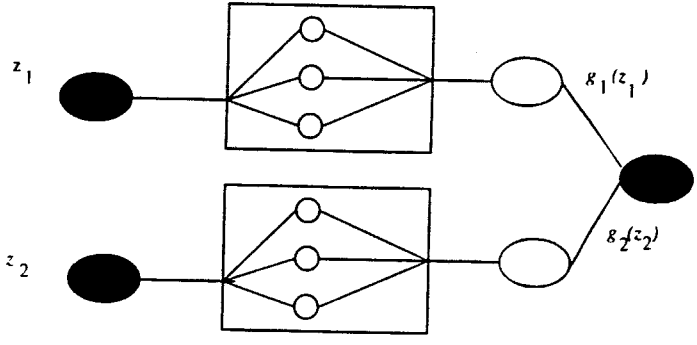


Figure 4. Additive Network

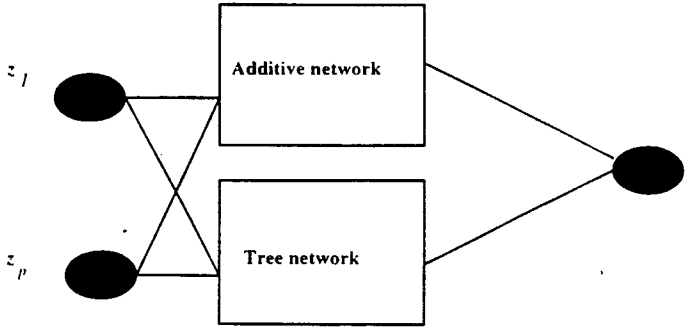


Figure 7. Network of networks.

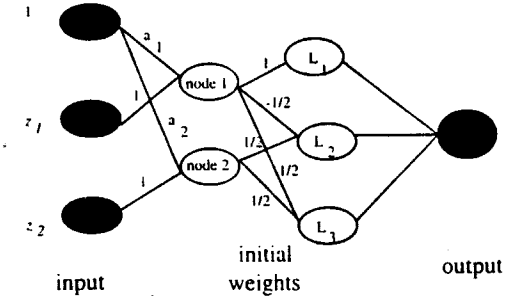


Figure 6. Tree Network

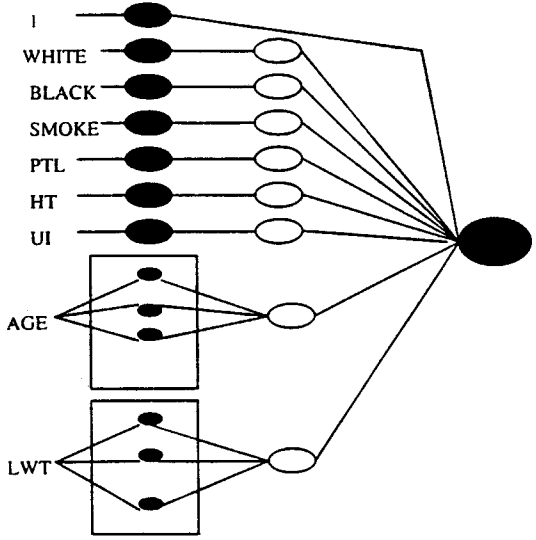


Figure 8. Additive Neural Network.

EXEMPLOS - CLASSIFICAÇÃO

1) Arminger, Enache e Bonne, 1997 comparam discriminação logística, regressão em árvores e redes neurais na análise de dados de risco em crédito.

A variável dependente é se o crédito é pago ou não. As variáveis preditoras são: sexo, tempo no emprego, estado civil, possuidor de carro e de telefone.

Os resultados são apresentados nas tabelas abaixo:

2) Desai, Crook e Overstreet, 1996 compararam análise de discriminantes linear, discriminação logística e redes neurais na análise de risco em crédito de três uniões de crédito. As redes utilizadas foram a logística e a de peritos locais. Os resultados são apresentados abaixo.

Tab. 2: Classification of the Cross Validation Sample using Logistic Discrimination

observed	predicted 0	1	
0	880	440	0.6667
1	446	968	0.6846
	1326	1408	0.6759

Tab. 7: Classification of the Test Sample using Logistic Discrimination

observed	predicted 0	1	
0	857	442	0.6597
1	460	986	0.6819
	1317	1428	0.6714

Tab. 5: Classification of the Cross Validation Sample using Classification Tree Analysis

observed	predicted 0	1	
0	882	438	0.6682
1	480	934	0.6605
	1362	1372	0.6642

Tab. 8: Classification of the Test Sample using Classification Tree Analysis

observed	predicted 0	1	
0	847	452	0.6520
1	488	958	0.6625
	1335	1410	0.6576

Tab. 6: Classification of the Cross Validation Sample using a Single-Hidden-Layer Feedforward Network

observed	predicted 0	1	
0	672	648	0.5091
1	302	1112	0.7864
	974	1760	0.6525

Tab. 9: Classification of the Test Sample using a Single-Hidden-Layer Feedforward Netwo

observed	predicted 0	1	
0	690	609	0.5312
1	332	1114	0.7704
	1022	1732	0.6572

Tab. 10: Allocation Table for the Combined Predictor

combination			class of y	
$\hat{y}_L$	$\hat{y}_T$	$\hat{y}_N$	0	1
0	0	0	**641**	259
1	0	0	**1**	1
0	1	0	30	**42**
1	1	0	**0**	0
0	0	1	**155**	110
1	0	1	85	**110**
0	1	1	**54**	34
1	1	1	354	**857**

Tab. 11: Classification for the Combined Predictor

	predicted		
observed	0	1	
0	826	473	0.6359
1	444	1002	0.6929
	1270	1475	0.6659

Table 1
List of predictor variables

MAJCARD	Number of major credit cards
OWNBUY	Owns home
INCOME	Salary plus other income
GOODCR	'Good' depending upon derogatory information and number of 01-09 ratings on credit bureau reports
JOBTIME	Number of years in current job
DEPENDS	Number of dependents
NUMINQ	Number of inquires in past 7 months
TRADEAGE	Number of months since trade line opened
TRADE75	Number of trade lines 75% full
PAYRATE1	Monthly payments as a proportion of income
DELNQ	Delinquent accounts in past 12 months
OLDDEBT	Total debt as a proportion of income
AGE	Age of borrower
ADDRTIME	Number of years at the current address
ACCTOPEN	Number of open accounts on credit bureau reports
ACTLOANS	Number of active accounts on credit bureau reports
PREVLOANS	Number of previous loans with credit union
DEROGINF	Information based upon 01-09 ratings on credit bureau reports

Model lda_c: Customized model using linear discriminant analysis.

Model lr_c: Customized model using logistic regression.

Model mlp_c: Customized model using multilayer perceptron.

Model lda_g: Generic model using linear discriminant analysis.

Model lr_g: Generic model using logistic regression.

Model mlp_g: Generic model using multilayer perceptron.

Model mnn_g: Generic model using modular neural network.

In the LDA models a case was classified into the group for which the posterior probability $P(G_j \mid Z)$ was greatest, where

$$P(G_j \mid Z) = \frac{P(Z \mid G_j)P(G_j)}{\sum_{i=1}^{2} P(Z \mid G_i)P(G_i)},$$

with:

$P(G_j) =$ Prior probability that a case is a member of group j.

$P(Z \mid G_j) =$ Conditional probability of a score of Z, given membership of group j.

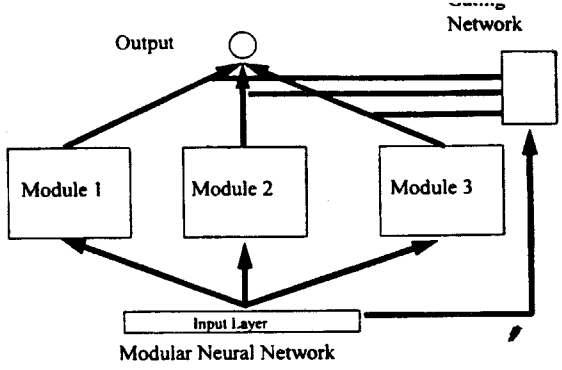


Fig. 2. Modular neural network.

Table 3
Generic models

Data set	Percentage correctly classified							
	mlp_g		mnn_g		lda_g		lr_g	
	% total	% bad	% total	% bad	% total	% bad	% total	% bad
1	79.97	28.29	80.12	28.95	79.80	27.63	80.30	33.55
2	82.26	35.06	81.80	38.31	80.60	31.82	81.40	35.06
3	81.04	36.44	79.20	47.46	82.40	37.29	82.70	40.68
4	82.88	33.80	83.33	38.03	83.49	40.85	83.49	44.37
5	79.21	32.88	78.59	43.15	80.60	37.67	80.01	39.04
6	81.04	53.69	80.73	35.57	80.58	38.93	81.35	42.95
7	80.73	44.30	79.82	44.97	80.73	36.24	82.57	41.61
8	78.89	50.00	79.82	41.96	80.58	32.88	81.35	39.73
9	81.92	43.87	80.43	32.90	81.04	30.97	81.35	33.55
10	79.51	62.50	80.73	32.64	81.35	33.33	82.42	40.28
average	80.75	42.08	80.46	38.39	81.12	34.76	81.70	39.08
p-value [a]			0.191	0.197	0.98	0.035	0.83	0.19

[a] The *p*-values are for a one-tailed paired *t*-test comparing the mlp_g results with the other three methods.

Customized models

Data set	Sample % bad	Percentage correctly classified					
		mlp_c		lda_c		lr_c	
		% total	% bad	% total	% bad	% total	% bad
Credit union L:							
1	14.88	83.33	32.00	83.30	16.00	82.70	24.00
2	17.26	87.50	31.03	82.70	20.69	81.00	34.48
3	17.26	86.90	41.38	83.90	31.03	82.10	37.93
4	22.02	81.55	37.84	84.52	32.43	82.74	37.84
5	18.45	82.14	29.03	77.40	16.13	78.00	38.71
6	17.26	83.33	41.38	81.55	27.59	82.74	41.38
7	21.43	80.36	16.67	78.50	08.33	80.95	30.56
8	17.86	79.76	56.67	82.14	10.00	81.55	23.33
9	16.67	85.71	32.14	86.31	32.14	86.90	39.29
10	18.45	83.33	29.03	79.76	19.35	80.36	29.03
Credit union M:							
1	26.77	85.43	79.71	86.60	70.97	87.40	73.53
2	26.77	88.58	76.47	85.40	18.64	88.20	79.41
3	20.47	85.43	80.77	87.40	31.58	85.80	75.00
4	23.63	90.94	75.00	90.20	35.14	89.00	75.00
5	24.02	89.37	88.52	89.80	22.22	88.60	81.97
6	26.38	88.58	74.63	88.19	12.96	86.22	71.64
7	26.77	85.43	74.63	85.04	28.30	86.22	77.61
8	25.59	88.19	75.38	86.61	28.89	87.40	67.69
9	27.59	87.40	72.86	85.43	31.37	86.61	67.14
10	24.41	90.55	82.26	89.37	21.05	89.37	80.65
Credit union N:							
1	25.43	75.86	49.15	75.40	18.64	78.50	27.11
2	24.57	77.15	40.35	7[illegible].50	31.58	78.50	24.56
3	15.95	81.90	35.14	83.20	35.14	84.10	35.14
4	19.40	77.15	44.44	78.90	22.22	80.60	28.89
5	23.28	76.72	24.07	75.40	12.96	75.90	18.52
6	22.84	79.31	35.85	75.00	28.30	76.72	26.42
7	19.40	81.90	31.11	80.60	28.89	81.03	28.89
8	21.98	74.13	37.25	74.57	31.37	75.43	31.37
9	24.57	80.17	47.37	77.59	21.05	78.88	21.05
10	21.98	77.59	19.61	77.16	21.57	76.72	19.61
average		83.19	49.72	82.35	38.49	82.67	44.93
p-value [a]				0.018	5.7E − 07	0.109	0.007

[a] The *p*-values are for a one-tailed paired *t*-test comparing the mlp_c results with the other two methods.

GRÁFICOS DE CONTROLE

A utilização de redes neurais em controle de processos foi estudada através de simulação por:

1) Prybutok, Sanford e Nam, 1994, compararam o tamanho médio da amostra (ARL), obtida com gráficos $\bar{X}$ e redes neurais para diversas regras de controle
   A rede usada foi: 5-5-1.
   A rede neural na maioria dos casos apresentou menor ARL.

2) Chen, 1997 apresenta extensiva simulação para controle de processos. A situação estudada incluia a rede da figura abaixo.

   A configuração foi:
   número de "expert network" - L = 3
   cada expert – 16 – 12 – 5
   "gating network" – 16 – 4 – 3

   A conclusão é que comparando com outros métodos de decisão automatizada o método tem vantagens.

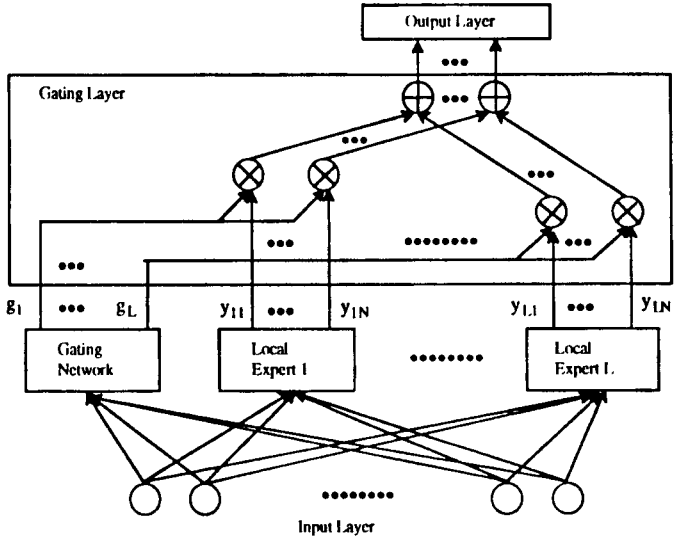


Figure 3. Network architecture.

Pattern name	Parameter	Comments
Trend	(0·2, 0·5, 0·025)	slope
Systematic variation	(1·0, 3·0, 0·25)	offset from the process mean
Cycle (period = 12)	(1·0, 3·0, 0·25)	amplitude
Mixture	(1·0, 3·0, 0·25)	offset from the process mean
Sudden shift	(1·0, 3·0, 0·25)	offset from the process mean

The notation (a, b, c) corresponds to (initial value, final value, increment).
The probability of shifting between distributions for mixtures = 0.4.

Table 1. Parameter values used in the training set.

Pattern	Desired output vector	Note
Upward trend	[1, 0, 0, 0, 0]	—
Downward trend	[−1, 0, 0, 0, 0]	—
Systematic variation (I)	[0, 1, 0, 0, 0]	the first observation is above the in-control mean
Systematic variation (II)	[0, −1, 0, 0, 0]	the first observation is below the in-control mean
Cycle (I)	[0, 0, 1, 0, 0]	sine wave
Cycle (II)	[0, 0, −1, 0, 0]	cosine wave
Mixture (I)	[0, 0, 0, 1, 0]	the mean of the first distribution is greater than the in-control mean
Mixture (II)	[0, 0, 0, −1, 0]	the mean of the first distribution is less than the in-control mean
Upward shift	[0, 0, 0, 0, 1]	—
Downward shift	[0, 0, 0, 0, −1]	—

Table 2. Desired output vectors.

SÉRIES TEMPORAIS

A utilização de redes neurais para previsão de séries temporais pode ser implementada por exemplo por:

1) com p-valores da série temporal como entrada, uma ou mais camadas escondidas e uma saída.

   A rede atua como uma autoregressão não-linear. Veja a rede abaixo:

2) Park, Murray e Chen, 1996 propõe escolher o número de neurônios da camada escondida, eliminando informação redundante, usando componentes principais nas saídas dos neurônios escondidos.

O procedimento é:

i) treinar a rede com grande número de neurônios escondidos

ii) obter a matriz de covariância das saídas dos neurônios escondidos

iii) tomar os autovalores maiores que um (p*)

iv) selecione os p* neurônios analisando as correlações entre os neurônios e as p* componentes principais

v) treine a nova rede.

3) Veiga e Medeiros, 1998 propõe um modelo ARX-N

$$y_t = a_0(t) + a_1(t)\, y_{t-1} + \ldots + a_p(t)\, y_{t-p} +$$

$$a_{p+1}(t)\, x_1(t) + \ldots + a_{p+q}(t)\, x_q(t) + \varepsilon_t$$

onde $a_j(t)$ são as saídas de uma rede neural, isto é:

$$a_t' = \left[a_0(t), a_1(t), \ldots a_p(t), \ldots, a_{p+q}(t)\right]$$

$$a_t' = NN(y_{t-1}, \ldots, y_{t-m}, x_1(t), \ldots, x_n(t))$$

A figura abaixo apresenta a rede correspodente.

O modelo é uma generalização do modelo TAR.

A estimação do modelo AR por redes neurais pode ser visto em Tian, Juhola e Gronfors, 1997 e em Veiga e Medeiros, 1998.

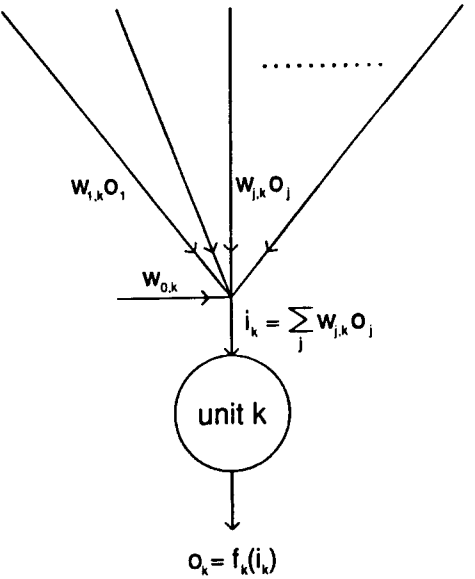


Figure 1. Neural Network Computation at a Single Unit.

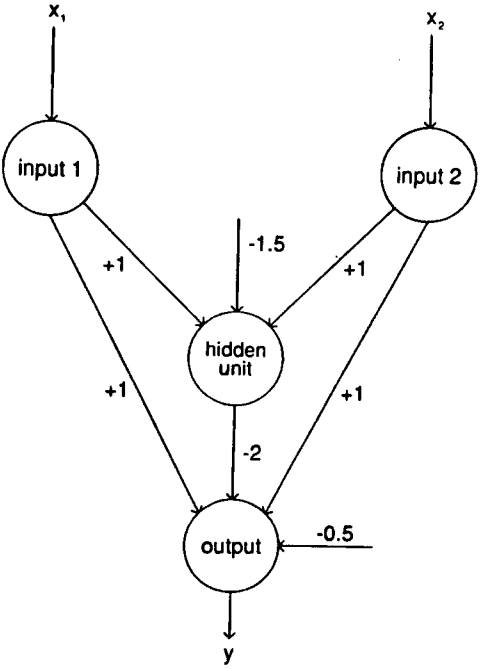


Figure 2. Neural Network for Exclusive-or. y = 1 if Exactly One of the Binary Inputs Is 1.

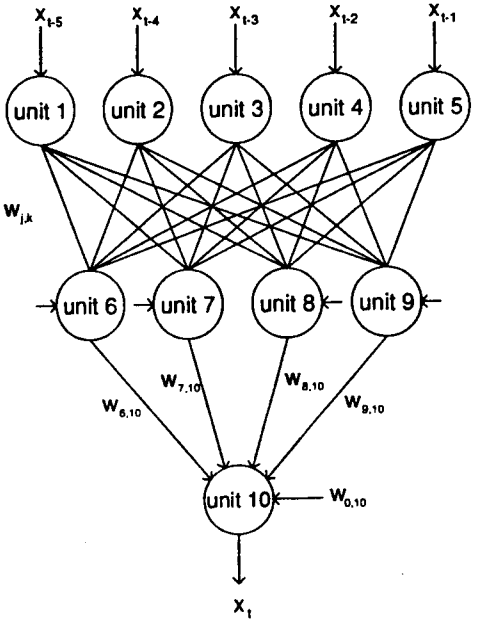


Figure 3. Neural Network for Time Series Prediction Problem.

EXEMPLOS – SÉRIES TEMPORAIS

Algumas aplicações de previsão com redes neurais são:

1) Raposo, 1992

2) Prykutok e Nam

3) Park, Murray e Chen, 1996

4) Veiga e Medeiros, 1998

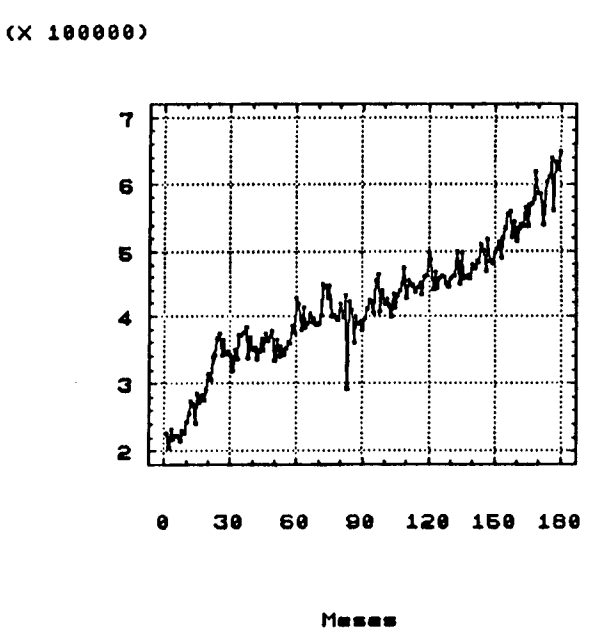


Figura V.6: Consumo de Energia de Ohio

$$z_t = z_{t-1} + z_{t-12} - z_{t-13} - 0.297a_{t-1} - 0.0704a_{t-12} + 0.209a_{t-13}$$

Tabela V.4:

Ano	*13 : 2 : 1*			*Box-Jenkins*
	Menor	Maior	Média	
1969	3.18 %	3.38 %	3.28 %	3.2 %
1970	3.72 %	4.20 %	4.00 %	4.18 %
Média	3.42 %	3.75 %	3.60 %	3.65 %

Tabela V.1:

Modelo	*Erro de Previsão*			*Treinamento*	
	Média	Menor	Maior	Erro	Método
Box-Jenkins	3.23 %	-	-	-	-
14:2:1	3.16 %	3.05 %	3.25 %	1.95 E-2	Aleatório
13:2:1	3.15 %	3.07 %	3.31 %	2.04 E-2	Aleatório
12:2:1	3.48 %	3.41 %	3.60 %	2.20 E-2	Aleatório

Tabela V.2:

Modelo	*Erro de Previsão*			*Treinamento*	
	Média	Menor	Maior	Erro	Método
Box-Jenkins	3.23 %	-	-	-	-
13:2:1	3.15 %	3.07 %	3.31 %	2.04 E-2	Aleatório
13:2:1	3.24 %	3.23 %	3.28 %	2.01 E-2	Normal
13:2:1	3.19 %	3.14 %	3.28 %	2.02 E-2	Grupo 10
13:2:1	3.25 %	3.19 %	3.29 %		Erro 2 %
13:2:1	3.21 %	3.09 %	3.30 %		Erro 2% e Rand.
13:2:1	3.26 %	3.20 %	3.34 %	2.00 E-2	Grupo 10 e Rand.

Tabela V.3:

Modelo	*Erro de Previsão*			*Treinamento*	
	Média	Menor	Maior	Erro	Método
Box-Jenkins	3.23 %	-	-	-	-
13:2:1	3.15 %	3.07 %	3.31 %	2.0 E-2	Randômico
13:2:1:1	3.19 %	3.11 %	3.31 %	2.0 E-2	Randômico

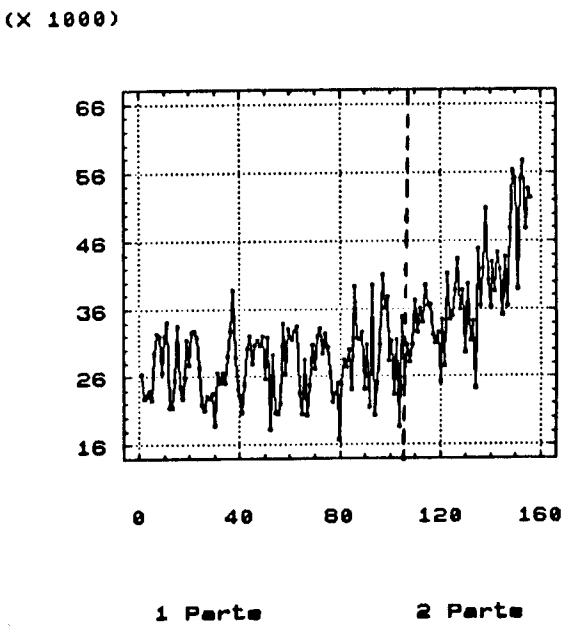


Figura V.9: Vendas de Equipamento Elétrico

$$(1 - \phi_1 B - \phi_2 B^2 - \phi_{12} B^{12})(Z_t - \mu) = a_t$$

Tabela V.5:

Ano	*13 : 2 : 1*			*Box-Jenkins*	
	Menor	Maior	Média	Original	"Logged"
1968	15.2 %	15.9 %	15.44 %	16.6 %	15.4 %
1969	13.1 %	13.7 %	13.38 %	11.5 %	10.5 %
1970	10.2 %	10.6 %	10.54 %	9.9 %	11.1 %
1971	11.8 %	12.3 %	12.0 %	14.1 %	11.7 %
1972	12.0 %	14.8 %	13.8 %	20.0 %	16.6 %
Média	12.72 %	13.28 %	13.03 %	14.42 %	13.06 %

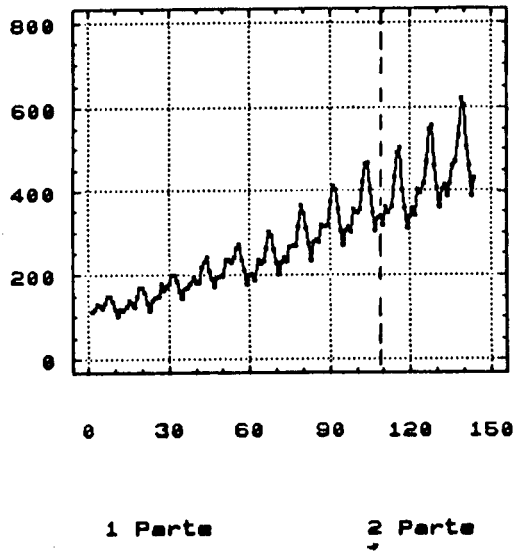


Figura V.12: Dados sobre Vôos Internacionais

$$(1-B)(1-B^{12})z_t = (1-\theta_1 B)(1-\theta_{12}B^{12})$$

Tabela V.6:

Ano	*13 : 2 : 1*			*Box-Jenkins*
	Menor	Maior	Média	
1958	6.05 %	6.96 %	6.44 %	7.75 %
1959	4.53 %	5.38 %	4.90 %	10.86 %
1960	7.29 %	8.60 %	7.93 %	15.55 %
Média	5.96 %	6.98 %	6.42 %	11.39 %

	Média			
Ano	*12 : 2 : 1*	*13 : 2 : 1*	*14 : 2 : 1*	*Box-Jenkins*
1958	7.63 %	6.44 %	5.00 %	7.75 %
1959	6.32 %	4.90 %	7.75 %	10.86 %
1960	7.39 %	7.93 %	11.91 %	15.55 %
Média	7.11 %	6.42 %	7.79 %	11.39 %

Table 1. Performance of neural networks for the different numbers of hidden neurons

Number of Hidden Neurons	MAD on Testing Data
1	25.36
3	27.76
6	**18.16**
12	20.57
20	18.18
30	19.12

Table 2. Performance of neural networks (12-6-1) for the different learning rates and momentums

Learning Rate	Momentum	MAD on Testing Data
0.1	0.0	18.25
0.1	0.9	18.16
0.6	0.0	21.63
0.6	**0.9**	**12.61**

** 12-6-1 means 12 inputs, 6 hidden neurons, and 1 output

Table 3. Effect of training time (12-6-1) for the different learning rates and momentums

Learning Rate	Momentum	Training Time of Network
0.1	0.0	11:53
0.1	0.9	02:10
0.6	0.0	06:37
0.6	**0.9**	**01:19**

** 00:00 means 00 minutes and 00 seconds

Table 4. MAD of testing data set

Month	Actual Value	Neural Network (12-6-1)	Exponential Smoothing	Linear Regression
Jan.	834	831.0	859.2	871.2
Feb.	782	794.1	858.9	834.5
Mar.	892	879.8	858.6	927.5
Apr.	903	898.2	858.4	944.5
May	966	962.6	858.1	1007.2
Jun.	937	937.8	857.8	980.7
Jul.	896	889.8	857.6	932.1
Aug.	858	842.7	857.3	891.3
Sep.	817	800.9	857.0	850.5
Oct.	827	802.1	856.8	855.3
Nov.	797	769.0	856.5	826.1
Dec.	843	826.5	856.2	864.0
MAD		12.6	45.7	36.1

Friedman test (p-value < 0.01)

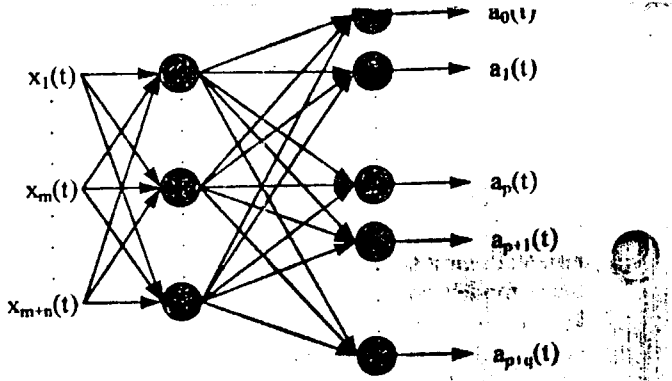


Figure 1: structure of the neural network.

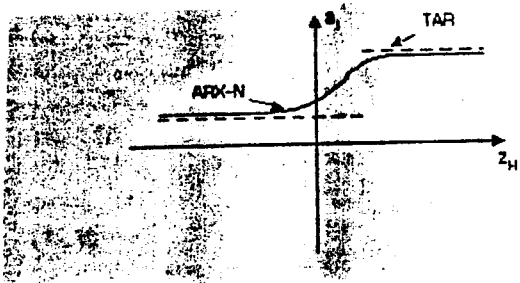


Figure 2: thresholds generated by an ARX-N model and TAR model.

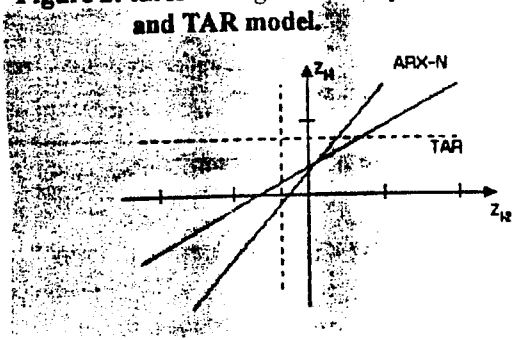


Figure 3: hyperplanes generated by the ARX-N model and the TAR model.

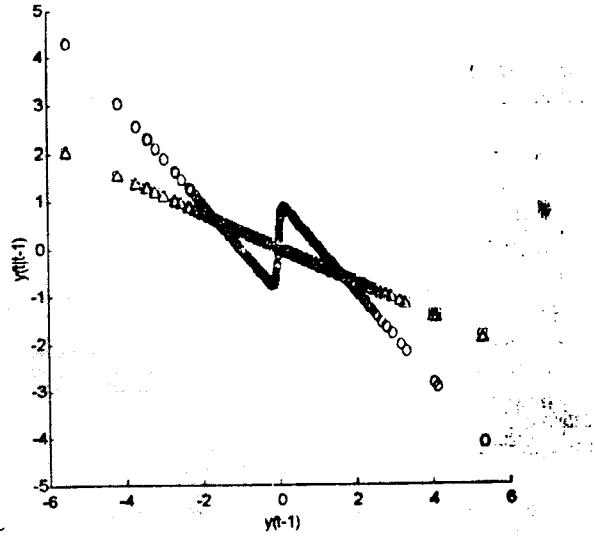


Figure 5: scatter plot y(t|t-1) x y(t-1). Circles: ARX-N;

Table 1: results of the three models.

	ARX-N	NARX	ARX
SSE (tranning)	539.41	708.62	708.10
mean(Error)(trainn.)	0.0002	0.0000	0.0000
var(Error) (trainning)	1.0832	1.4229	1.4218
nRMSE (trainning)	0.8117	0.9304	0.9300
U-Theil (trainning)	0.4918	0.5637	0.5635
SSE (forecast)	531.31	727.33	717.10
mean(Error) (forecast)	-0.0729	0.0054	0.0000
var(Error) (forecast)	1.0594	1.4575	1.4351
nRMSE (forecast)	0.8091	0.9466	0.9400
U-Theil (forecast)	0.4949	0.5790	0.5750

Table 2: results of the three models.

	ARX-N	NARX	ARX
SSE (tranning)	513.60	516.86	531.80
mean(Error) (trainn.)	-0.0216	0.0055	-0.100
var(Error) (trainning)	1.0329	1.0399	1.0600
nRMSE (trainning)	0.9317	0.9346	0.9481
U-Theil (trainning)	0.5837	0.5855	0.5838
SSE (forecast)	480.00	480.60	486.55
mean(Error) (forecast)	-0.0623	-0.032	-0.131
var(Error) (forecast)	0.9580	0.9620	0.9579
nRMSE (forecast)	0.9659	0.9665	0.9725
U-Theil (forecast)	0.6069	0.6072	0.6110

Table 3: models used to forecast the laser series.

p	S_{ARX-N}=2,4,6,...,20	$S_{RN} = \left[\frac{2(p+1)S_{ARX-N}+p}{p+2}\right]$	n/a
2	ARX-N(2,S_{ARX-N},3)	NARX(2,S_{RN},1)	AR(2)
4	ARX-N(4,S_{ARX-N},5)	NARX(4,S_{RN},1)	AR(4)
6	ARX-N(6,S_{ARX-N},7)	NARX(6,S_{RN},1)	AR(6)
8	ARX-N(8,S_{ARX-N},9)	NARX(8,S_{RN},1)	AR(8)
10	ARX-N(10,S_{ARX-N},11)	NARX(10,S_{RN},1)	AR(10)

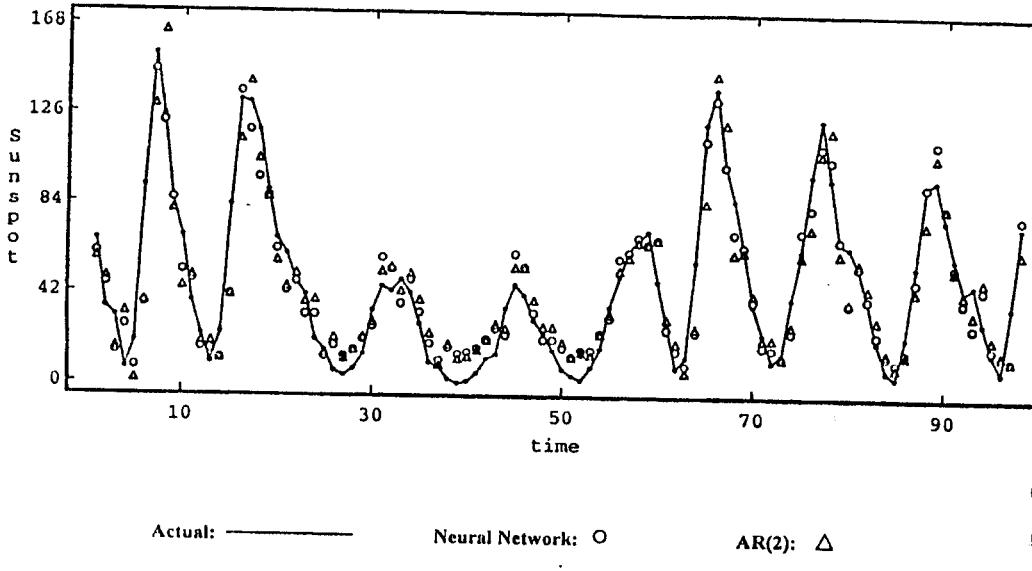


Fig. 1. Sunspot data (1770–1869).

TABLE III

ARY OF FORECASTING ERRORS PRODUCED BY NEURAL NETWORK MODELS AND THE AR(2) MODEL BASED ON UNTRAINED D

Model	Mean Square Error (MSE)	Mean Absolute Error (MAE)
2 x 11 x 1	160.8409	9.2631
2 x 3 x 1	158.9064	9.1710
2 x 2 x 1*	157.7880	9.1522
2 x 1 x 1	180.7608	11.0378
2 x 0 x 1	180.4559	12.1861
AR(2)	177.2645	11.5624

* represents the selected model by the PCA method.

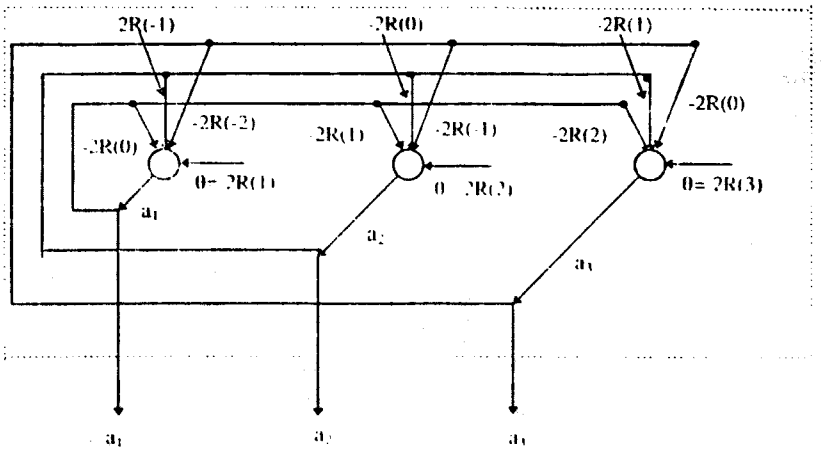


Fig. 1. Hardware scheme of a feedback artificial neural network that can estimate AR parameters. The solutions can be considered as stable state of the dynamic system.

Table 2
AR parameters estimated by Levinson–Durbin method (LD) and a feedback neural network method with iterative number of 1000 in case of AR model of order 6 and 2 for the ABR signal, respectively

	Order = 6						Order = 2	
	a_1	a_2	a_3	a_4	a_5	a_6	a_1	a_2
LD	−1.2976	0.1355	0.0466	0.0322	0.0360	0.0629	−1.5234	0.5321
ANN	−1.2940	0.1319	0.0478	0.0308	0.0354	0.0636	−1.5234	0.5320

TESTES DE NÃO-LINEARIDADE

Os testes de não-linearidades são aplicados às séries temporais em que se desconfie haver relações não-lineares entre as variáveis. Estes testes são baseados na hipótese de haver linearidade contra a hipótese alternativa de não haver linearidade entre as variáveis. Este fato é muito encontrado em séries temporais econômicas.

Considere $\{Z_e\}$ um processo estocástico e a partição Z_e como $Z_e = (y_t, X_t)'$, onde y_t é um escalar e X_t um vetor k x 1. X_t pode conter uma constante e valores atrasados de y_t. O processo (y_t) é linear em média condicional a x_t se

$$P\left[E(y_t/X_t) = X_t' \theta^*\right] = 1 \text{ para algum } \theta^* \in \Re^k$$

A hipótese alternativa é de y_t não seja linear em média condicional a X_t, assim:

$$P\left[E(y_t/X_t) = X_t' \theta\right] < 1 \quad \text{para todo } \theta \in \Re^k$$

Quando esta hipótese for verdadeira diremos que o modelo linear sofre de uma não-linearidade negligenciada.

O teste de RN para não-linearidade negligenciada utiliza uma rede com uma camada escondida. A saída da rede é então

$$0 = x\theta + \sum_{j=1}^{0} \beta_j \psi(x\gamma_j)$$

Quando a hipótese nula de não-linearidade for verdadeira, i.e.,

$$H_e: \; P\left[E(y_t X_t) = X_t' \theta^*\right] = 1 \quad \text{para algum } \theta_t^*$$

Então, os pesos ótimos da rede β_j, β^*_j, são zeros, j = 1, ..., q, fornecendo uma rede "affine". O teste da RN de não-linearidade negligenciada testa a hipótese $\beta^*_j = 0$, j = 1, ..., q, para uma escolha particular de q e γ_j. O teste terá potência sempre que

$$\sum_{j=1} \beta_j \psi(x' y_j)$$

for capaz de extrair estrutura de

$$e_t^* = y_t - X_t' \theta^*$$

Sob a alternativa, θ* é o vetor parâmetro da aproximação de MQ linear ótima para

$$E(y_t / X_t)$$

Implementando o teste como um teste do multiplicador de Lagrange, temos:

$$H_0' : E(\psi_t \, e_t^*) = 0$$

$$H_a' : E(\psi_t, e_t^*) \neq 0$$

Onde

$$\psi_t \equiv \left(\psi(X_t', \Gamma_1), \cdots, \psi(X_t', \Gamma_q)\right)', \text{ e } \Gamma = (\Gamma_1, ..., \Gamma_q)$$

são escolhidos a priori, independentemente da sequência aleatória $[X_t]$, para um dado $q \in N$, Γ é escolhido aleatoriamente.

Para construir o teste, substituimos e^*_t pelos resíduos estimados $e_t = y_t - X'_t \theta$, com θ obtido pelos MQ. Então:

$$M_n = \left(n^{-1/2} \sum_{t=1}^{n} \psi_t \hat{e}_t' \right)' \overline{W}_n^{-1} \left(n^{-1/2} \sum_{t=1}^{n} \psi_t \hat{e}_t \right)$$

Onde W_n é um estimador consistente de $W^* = \text{var}(n^{-1/2} \Sigma \psi_t e_t)$. Assintoticamente, chega-se a conclusão de que $M_t \rightarrow \chi^2(q)$ quando $n \rightarrow \infty$ sob H_0.

Devido a tendência dos elementos de ψ_t serem colineares com X_t e com eles próprios, podemos conduzir o teste usando $q^* < q$ componentes principais de ψ_t não colineares com X_t, denotados de

ψ_{t*}. E, devido também a vagarosidade do cálculo de W_s, devemos utilizar um teste estatístico equivalente que evite o cálculo formal de W_s, ou seja,

$$nR^2 \xrightarrow{d} \chi^2(q*)$$

onde R^2 é a correlação múltipla quadrática não-centralizada da regressão linear padrão de e_t em ψ_t*, X_t.

EXEMPLO – TESTE DE NÃO LINEARIDADE

Lee et al (1993) gerou grupos, várias séries univariadas para diferentes situações não-lineares com o objetivo de comparar o teste de rede neural de não-linearidade negligenciada com outros testes alternativos, tais como, o de Keenan, Tsay, White, McLeod e BDS. A função de ativação utilizada foi a logística

$$\psi(\lambda)=[1+\exp(-\lambda)]^{-1}$$

A entrada para os pesos da camada escondida Γ_{ij} foram gerados aleatoriamente da distribuição uniforme [-2, 2]. As variáveis y_t, X_t escalonada em [0, 1]. Para NEURAL1 e para NEURAL2 (evitando colinearidade) escolherem $q = 10$ e $q = 30$, respectivamente. Usou $q^* = 2$, isto é, os dois maiores componentes principais (excluindo o primeiro componente principal) para um grupo de modelos (grupo 1), e modelos bivariados e $q^* = 3$ para os modelos do grupo 2.

Para o cálculo dos testes as séries foram transformadas para produzir sequências estacionárias. A série temporal é ajustada a um modelo AR(p), onde p é determinado pelo critério SIC.

Efeitos ARCH não foram investigados. Os modelos são:

Grupo 1: AR, BL, TAR, SGN, NAR

Grupo 2: MA, MAheteroscedastic, NMA, AR(2), BAR, BARMA, Bivariate (SQ, EXP)

Os resultados indicaram os testes usando redes neurais foram tão potentes ou mais potentes que os testes alternativos.

REFERÊNCIAS

BIBLIOGRAFIA

Anderson, J. A. – 1995 – An Introduction to Neural Networls MIT Press.

Arminger, G. e Enache, D. – 1996 – Statistical models and artificial neural networks. Em Data Analysis and Information Systems, (Eds. H. H. Bock e W. Polasek), V. 7, 243-260.

Arminger, G., Enache, D. e Bonne, T. – 1997 – Analysing credit risk data: a comparison of logistic descrimination, classification tree analysis and feedfoward networks. Computational Statistics, V. 12, 293-310.

Bigangoli, E., Baracchi, P., Mariani, L. e Marubini, E. – 1998 – Feed forward neural networks for the analysis of censored survival data: a partial logistic regression approach. Statistics in Medicine, V. 17, 1169-1186.

Carvalho, L. A. V. – 1994 – Modeling the thalamocortical loop. International Journal of Bio-Medical Computing, V. 35, 267-296.

Carvalho, L. A. V. e Diego, A. C. M. – 1997 – Positive feedback loop between norepinephrine and corticotrapin-releasing factor-containing pontine nucley-may subserve chronic stress. Manuscrito COPPE/UFRJ.

Carvalho, L. A. V. – 1998 – Teoria da mente: a alma humana em busca de si mesma. Em Descartes: Um Legado Científico e Filosófico (Ed. S. Fuks), Relume Dumara Editora.

Cheng, B. e Tittermgton, D. M. – 1994 – Neural networks: a review from a statistical perspective (with discussion). Statistical Sciences, V. 9, 2-54.

Cheng, C. S. – 1997 – A neural network approach for the analysis of control chart pattersn. International Journal of Production Research, V. 35, 667-697.

Chryssolouris, G., Lee, M. e Ramsey, A. – 1996 – Confidence interval prediction for neural network models. IEEE Transactions on Neural Networks, V. 7, 229-232.

Church, K. B. e Curram, S. P. – 1996 – Forecasting consumer's expenditure: a comparison between econometric and neural network models. International Journal of Forecasting, V. 12, 255-267.

Ciampi, A. e Lechevallier, Y. – 1997 – Statistical models as building blocks of neural networks. Communications in Statistics –Theory, Methods, V. 26, 991-1009.

Desai, V., Crook, J. N. e Overstreet Jr., G. A. – 1996 – A comparison of neural networks and linear scoring models in the credit union environment. European Journal of Operational Research, V. 95, 24-37.

Faraggi, D. e Simon, R. – 1995 – A neural network for survival data. Statistics in Medicine, V. 14, 73-82.

Faraway, I. e Chatfield, C. – 1998 – Time series forecasting with neural networks a comparative study using the airline data. Applied Statistics, V. 47, 231-250.

Harvey, A. e Toulson, S. – 1994 – Review of '4 Through'. International Journal of Forecasting, V. 10, 35-41.

Haykin, S. – 1994 – Neural networks a Comprehensive Foundation. MacMillan/

Hill, T., Marquez, L., Oconnor, M. e Remus, W. – 1994 – Artificial neural network models for forecasting and decision making. International Journal of Forecasting, V. 10, 5-15.

Kovacs, I. H. – 1997 – O cérebro e a sua mente: uma introdução a neurociência computacional. Edição Academica.

Kovacs, Z. L. – 1996 – Redes Neurais Artificiais. Edição Academica.

Lee, T. H., White, H. e Granger, C. W. J. – 1993 – Testing for negleted nonlinearity in time series models: a comparison of neural network methods and alternative tests. Journal of Econometrics, V. 56, 269-290.

Liestol, K., Andersen, P. K. e Andersen, U. – 1994 – Survival analysis and neural nets. Statistics in Medicine, V. 13, 1189-1200.

Lourenço, P. M. – 1998 – Um modelo de previsão de curto prazo de carga elétrica combinando métodos estatísticos e inteligência computacional. Tese de Doutorado, Engenharia Elétrica, PUC/RJ.

Manguameli, P., Chen, S. K. e West, D. – 1996 – A comparison of SOM neural networks and hierarchical clustering methods. European Journal of Operation Research, V. 93, 402-417.

Murtagh, F. – 1994 – Neural networks and related "massively parallel" methods for statistics: a short review. International Statistical Review, V. 62, 275-288.

Park, Y. R., Murray, T. J. e Chen, C. – 1996 – Predicting sun spots using layered perceptron neural networks, IEEE Transactions on Neural Networks, V. 7, 501-505.

Poli, J. e Jones, R. D. – 1994 – A neural net model for prediction. Journal of the American Statistical Association, V. 89, 117-121.

Prybutok, V. K., Sanford, C. C. e Nam, K. T. – 1994 – Comparison of neural networks to Shewhart X control chart applications. Economic Quality Control, V. 9, 143-164.

Raposo, C. A. – 1992 – Redes neuronais na previsão em séries temporais. Tese de Mestrado, Engenharia de Sistemas e Computação, COPPE/UFRJ.

Ripley, B. D. – 1994 – Neural networks and related methods for classification (with discussion). Journal of the Royal Statistical Society, V. 56, 409-456.

Ripley, B. D. – 1994 – Neural networks and flexible regression and discrimination. Advances in Applied Statistics: Statistics and Images. A Supplement to Journal of Applied Statistics, V. 21.

Ripley, B. D. – 1996 – Pattern Recognition and Neural Networks. Cambridge University Press.

Rosenthal, M., Engelhard, E. e Carvalho, L. A. V. – 1998 – Adaptive resonance theory in the search of neropsychological pattersn of schizophrenia. Manuscrito COPPE e IPUB, UFRJ.

Schumacher, M.; Robner, R. e Vach, W. – 1996 – Neural networks and logistic regression: Part I. Computational Statistics and Data Analysis, V. 21, 661-682.

Smith, M. – 1993 – Neural network for statistical modelling, Van Nostrond, Reinhold.

Stern, H. S. – 1996 – Neural networks in applied statistics (with discussion). Technometries, V. 38, 205-220.

Tian, J., Johola, M. e Gronfors, T. – 1997 – AR parameter estimation by a feedback neural network. Computational Statistics and Data Analysis, V. 25, 17-24.

Tibshirani, R. – 1996 – A comparison of some error estimates for neural networks models. Neural Computation, V. 8, 152-163.

Vach, W., Robner, R. e Schumacher, M. – 1996 – Neural networks and logistic regression: Part II. Computational Statistics and Data Analysis, V. 21, 683-701.

Veiga, A. e Medeiros, M. C. – 1998 – An hybrid linear-neural model for financial time series. V International Conference on Forecasting Finantial Markets.

Warner, B. e Misra, M. – 1996 – Understanding neural networks as statistical tools American Statistician, V. 50, 281-293.

Weiss, S. M. e Kulikowski, C. A. – 1990 – Computer Systems that Learn. Morgan Kanfmann Publishers.