

---

# ANÁLISE COMPARATIVA DE DESEMPENHO DE MEMÓRIAS ASSOCIATIVAS EM REDES NEURAIS CELULARES

**Leonardo Garcia Corrêa\***  
lcorre@icmc.usp.br

**Alexandre C B Delbem\***  
acbd@icmc.usp.br

**Liang Zhao\***  
zhao@icmc.usp.br

\*Instituto de Ciências Matemáticas e de Computação - ICMC  
Universidade de São Paulo - USP  
Caixa Postal 668, 13560-970 São Carlos, SP, Brasil

---

## ABSTRACT

This paper presents a performance comparison of various methods for associative memory design in Cellular Neural Networks (CNNs). Even though there have been an increasing interest in such kind of application for CNNs, there is no comprehensive comparison of the performance among the proposed methods in the literature. In this paper, existing design methods are reviewed and their corresponding performances are measured in numerical simulations. The T-Student test is then employed in order to compare the results obtained.

**KEYWORDS:** Cellular Neural Networks, Associative Memory, Performance Comparison.

## RESUMO

Este artigo apresenta uma comparação de desempenho de vários métodos para o projeto de memórias associativas com Redes Neurais Celulares (RNCs). Embora tenha ocorrido um interesse crescente em tal tipo de aplicação de RNCs, não há uma comparação compreensível do desempenho entre os métodos propostos na literatura. Neste artigo, métodos existentes para projeto são revisados e o desempenho deles

é medido por simulações numéricas. O teste T de Student é então utilizado para comparar os resultados obtidos.

**PALAVRAS-CHAVE:** Redes Neurais Celulares, Memória Associativa, Comparação de Desempenho.

## 1 INTRODUÇÃO

As Redes Neurais Celulares (RNCs) são redes localmente acopladas ou dotadas de matrizes de conexões esparsas. Tais redes têm recebido crescente atenção uma vez que diversos sistemas biológicos podem ser modelados como sistemas dinâmicos no espaço e no tempo com elementos que interagem apenas localmente (Kaneko, 1989a; Kaneko, 1989b; Wolfram, 1983).

Além disso, as RNCs podem ser adequadamente implementadas em hardware pois seus neurônios requerem apenas conexões com os neurônios mais próximos. Essa característica torna as RNCs atrativas para aplicações como processamento de imagens (Crouse and Chua, 1995), memória associativa e reconhecimento de padrões (Michel and Farell, 1990), entre outras (Thiran et al., 1995).

As RNCs, idealizadas em (Chua and Yang, 1988), são redes neurais realimentadas de células (ou neurônios) conectadas apenas localmente. O nome RNC advém da semelhança com Autômatos Celulares (Wolfram, 1986) quanto ao padrão local de interconexão entre as células da rede. A literatura na área tem demonstrado inúmeras aplicações dessas

---

Artigo submetido em 11/01/2007

1a. Revisão em 07/07/2007

2a. Revisão em 01/07/2008

Aceito sob recomendação do Editor Associado  
Prof. Ivan Nunes da Silva

redes principalmente em visão computacional (Werblin et al., 1995), simulação numérica de Equações Diferenciais Parciais (EDPs) (Roska et al., 1995), (Kozek et al., 1995) e memória associativa (Liu and Michel, 1994), (Liu and Lu, 1997), (Park et al., 2001), (Szolgay et al., 1997), (Grassi, 2001).

Propriedades computacionais coletivas emergentes, como as exibidas por memórias associativas em redes neurais realimentadas, têm sido investigadas principalmente após o trabalho clássico de Hopfield (Hopfield, 1982). Devido às vantagens de implementação em hardware apontadas para as RNCs, tais propriedades têm sido também exploradas nessas últimas. Nesse artigo, serão apresentados diversos métodos propostos na literatura para a construção de memórias associativas em RNCs, que serão avaliados na tarefa de reconhecimento de padrões em imagens binárias.

Memórias associativas fazem referência à classe de memórias endereçáveis pelo conteúdo da informação armazenada, apresentando, portanto, certa plausibilidade biológica com as características de memória exibidas pelo cérebro. Em contraste com o modelo de memória implementado por um computador digital, que endereça informações com base na localização física das mesmas, as memórias associativas armazenam as informações de forma distribuída na rede. Além disso, o processo de recuperação de informações consiste no relaxamento das equações que descrevem a dinâmica da rede, cuja solução normalmente converge para um atrator (no caso estudado, um ponto de equilíbrio), que representa a informação procurada. Uma das vantagens dessa estratégia é a rapidez na busca pelas informações, o que ocorre de forma não-exaustiva, além da robustez resultante do armazenamento distribuído das mesmas.

Além da rapidez, os experimentos mostram que algumas RNCs, como as baseadas em LMIs (*Linear Matrix Inequalities*) e aprendizado Hebbiano podem conseguir taxas de recuperação correta de padrões significativamente superiores às da rede de Hopfield. Este comportamento mostrou-se mais enfático para os casos em que a carga (número de padrões armazenados) aumentava consideravelmente.

Diversos métodos para construção de memórias associativas em RNCs têm sido propostos na literatura. Entretanto, os resultados disponíveis não permitem comparar o desempenho das memórias obtidas por cada um desses métodos. O presente artigo apresenta uma análise comparativa criteriosa e sistemática do desempenho dos diversos métodos propostos, além de uma revisão destes.

A Seção 2 apresenta a revisão dos diversos métodos encontrados na literatura para construção de memórias associativas em RNCs. A Seção 3 descreve os experimentos realizados com os diferentes métodos apresentados. A Seção 4 resume os resultados obtidos com os experimentos, por meio de grá-

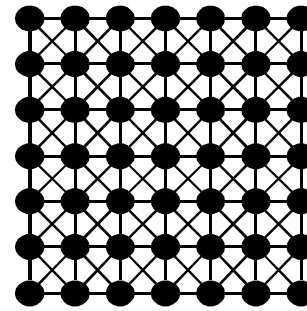


Figura 1: Representação de uma RNC. Os nós denotam células e as arestas denotam interconexões.

ficos e de um teste estatístico (teste T de Student). Finalmente, a Seção 5 conclui o artigo com uma breve discussão.

## 2 MEMÓRIA ASSOCIATIVA EM RNC

Uma RNC pode ser vista como um arranjo bidimensional de células (Figura 1) interconectadas apenas localmente. A vizinhança de uma célula  $c(i, j)$  é denotada por

$$V_r(i, j) = \{c(k, l) : \max(|k - i|, |l - j|) \leq r\}, \quad (1)$$

$$1 \leq k \leq M, \quad 1 \leq l \leq N.$$

onde  $r$  indica o raio da vizinhança em torno da célula e  $M \times N$  a quantidade total de células da rede.

A operação de uma RNC é descrita pela seguinte equação diferencial:

$$\begin{aligned} \dot{x} &= -x + Ty + I \\ y &= \text{sat}(x) \end{aligned} \quad (2)$$

$$\text{sat}(x_i) = \begin{cases} -1, & \text{se } x_i < -1 \\ x_i, & \text{se } -1 \leq x_i \leq 1, \\ 1, & \text{se } x_i > 1 \end{cases} \quad (3)$$

$$i = 1, \dots, n.$$

onde  $x(t) \in \mathbb{R}^n$  denota o estado das células,  $T \in \mathbb{R}^{n \times n}$  representa a matriz (esparsa) de interconexões,  $y \in D^n$  ( $D^n = \{y \in \mathbb{R}^n : -1 \leq y_i \leq 1, i = 1, \dots, n\}$ ) representa a saída (bipolar) das células,  $\text{sat}(x) = [\text{sat}(x_1), \dots, \text{sat}(x_n)]^T$  é a função saturação (equação (3)),  $I \in \mathbb{R}^n$  é um vetor de *bias* e  $n = M \times N$  representa o número de células da rede.

A fim de se construir uma memória associativa em uma RNC, os pontos de equilíbrio de (2) precisam ser ajustados para que a saída (binária ou bipolar)  $y$  da rede corresponda aos

diferentes padrões que se deseje armazenar. Para isso, tal equação é resolvida em termos dos parâmetros  $T$  e  $I$ , pelos diversos métodos apresentados nas seções que seguem.

## 2.1 Método baseado em matrizes Pseudo-Inversas

Em redes de Hopfield, os dados de entrada são fornecidos por meio das condições iniciais; as saídas atingem o estado de regime para o ponto de equilíbrio, que depende da condição inicial. Em RNCs, o ponto de equilíbrio pode independe do valor inicial de  $X$ . Para tanto, deve-se escolher um relação adequada entre  $X$  e  $Y$  e alimentar a rede por  $Y$ . Com isso, um hardware implementando essa solução de RNC não precisa ter seus estados reinicializados a cada nova execução.

O método proposto em (Grassi, 2001) utiliza matrizes pseudo-inversas para resolver as equações de equilíbrio de (2):

$$\begin{aligned} x^p &= Ty^p + I, \quad p = 1, \dots, m \\ y^p &= \text{sat}(x^p) \end{aligned} \quad (4)$$

onde  $y^p \in B^n$  ( $B^n = \{-1, 1\}^n$ ) representa os  $p$  padrões a serem armazenados e  $x^p \in C^n(y^p)$  ( $C^n(y) = \{x \in \mathbb{R}^n : x_i y_i > 1, i = 1, \dots, n\}, \forall y \in B^n$ ) os pontos de equilíbrios associados. A estratégia considerada para solução da equação acima consiste em reescrever (4) em sua forma matricial, para depois isolar as variáveis  $T$  e  $I$  que devem ser determinadas. Assim, de (4):

$$X = TY + \bar{I}, \quad (5)$$

onde  $Y = [y^1, \dots, y^m] \in \mathbb{R}^{n \times m}$ ,  $X = \alpha Y \in \mathbb{R}^{n \times m}$ ,  $\alpha > 1$  e  $\bar{I} = [I, \dots, I] \in \mathbb{R}^{n \times m}$ . A escolha de  $X = \alpha Y$ , para algum  $\alpha > 1$ , faz com que os pontos de equilíbrio existam apenas dentro da região de saturação ( $|x_i^p| > 1$ ), o que garante maior robustez na recuperação dos padrões (Liu and Michel, 1994).

Dado que a matriz  $T$  em (5) é esparsa (devido à RNC ser uma rede de células acopladas apenas localmente), essa equação precisa ser reescrita levando em conta tal restrição. Como o número de elementos em cada linha da matriz  $T$  é igual a zero, por não existirem conexões entre células fora da vizinhança da  $i$ -ésima célula, esses elementos podem ser eliminados de (5), juntamente com os respectivos componentes de  $Y$  envolvidos na multiplicação. Seja  $\tilde{T}_i \in \mathbb{R}^{1 \times v_i}$  o vetor resultante da eliminação dos elementos iguais a zeros da  $i$ -ésima linha de  $T$  (denotada  $T_i$ ), e seja  $\tilde{Y}_i \in \mathbb{R}^{v_i \times m}$  o vetor resultante da eliminação dos elementos que são multiplicados por posições nulas de  $T_i$ , onde  $v_i$  indica a quantidade de

vizinhos da  $i$ -ésima célula (ver (Liu and Michel, 1994) para maiores detalhes). Então, para cada célula  $i = 1, \dots, n$  da rede, tem-se, de (5):

$$X_i = \tilde{T}_i \tilde{Y}_i + \bar{I}_i, \quad i = 1, \dots, n \quad (6)$$

onde  $X_i \in \mathbb{R}^{1 \times m}$  e  $\bar{I}_i \in \mathbb{R}^{1 \times m}$  denotam as  $i$ -ésimas linhas de  $X$  e  $\bar{I}$ , respectivamente.

Para isolar as variáveis  $\tilde{T}_i$  e  $\tilde{Y}_i$  que precisam ser determinadas da equação anterior, definem-se

$$\begin{aligned} \tilde{R}_i &= [\tilde{Y}_i^T, J] \in \mathbb{R}^{m \times (v_i+1)}, \\ J &= [1, \dots, 1]^T \in \mathbb{R}^m, \\ \tilde{W}_i &= [\tilde{T}_i, I_i] \in \mathbb{R}^{1 \times (v_i+1)}. \end{aligned}$$

Assim, de (6) resulta

$$X_i^T = \tilde{R}_i \tilde{W}_i^T, \quad i = 1, \dots, n$$

Esta equação possui solução desde que  $\tilde{R}_i$  seja invertível. Como nem sempre é esse o caso, utiliza-se a seguinte solução:

$$\hat{W}_i^T = \tilde{R}_i^+ X_i^T, \quad i = 1, \dots, n \quad (7)$$

onde  $\tilde{R}_i^+$  denota a matriz pseudo-inversa (Israel and Greenville, 1974) de  $\tilde{R}_i$ , que sempre pode ser obtida. Essa solução tem a propriedade de minimizar norma quadrática do erro de reconstrução (Lay, 1996):

$$\|X_i^T - \tilde{R}_i \hat{W}_i^T\|^2, \quad i = 1, \dots, n.$$

onde  $\|\cdot\|$  denota a norma Euclidiana.

**Observação 2.1** Para se reconstruir a matriz  $T$ , a partir da solução  $\hat{W}_i$  em (7), os  $v_i$  componentes de  $\hat{W}_i$  são reatribuídos às suas respectivas posições na  $i$ -ésima linha da matriz  $T$ , de acordo com a posição dos vizinhos da  $i$ -ésima célula. O último componente do vetor  $\hat{W}_i$  corresponde ao valor do parâmetro  $I_i$ , para essa mesma  $i$ -ésima célula. Os elementos da matriz  $T$  que não tiveram um valor atribuído, por não fazerem referência à nenhuma das vizinhanças das células da rede, são igualados a zero.

O método para determinação dos parâmetros da RNC, descrito nessa seção, é sintetizado pelo Algoritmo 2.1.

**Algoritmo 2.1** Algoritmo de treinamento baseado em matrizes pseudo-inversas

**Require:**  $m$  padrões de treinamento  $y^p, p = 1, \dots, m$

**Ensure:** parâmetros  $T, I$  da RNC

Construa as matrizes  $Y = [y^1, y^2, \dots, y^m]$  e  $X = \alpha Y$

**for**  $i = 1, \dots, n$  **do**

Construa  $\tilde{T}_i$  e  $\tilde{Y}_i$  a partir de  $T$  e  $Y$ . {ver instruções no texto}

Construa

$$\begin{aligned}\tilde{R}_i &= [\tilde{Y}_i^T, J] \\ J &= [1, \dots, 1]^T \\ \tilde{W}_i &= [\tilde{T}_i, I_i]\end{aligned}$$

Calcule

$$\hat{W}_i^T = \tilde{R}_i^+ X_i^T$$

Reconstrua  $T$  e  $I$  a partir de  $\hat{W}_i$ . {ver instruções na Observação 2.1}

**end for**

## 2.2 Método baseado em Decomposição por Valor Singular (SVD)

A estratégia de RNCs utilizando Decomposição por Valor Singular (SVD), assim como o método baseado em matrizes pseudo-inversas, também possibilita soluções independentes do estado inicial da rede. Além disso, por meio de SVD, pode-se verificar se há estados espúrios (estados de equilíbrio que não correspondem a um padrão do treinamento da rede) em uma RNC e desenvolver estratégias para evitar tais estados.

O método baseado na fatoração de matrizes, denominada Decomposição por Valor Singular (Lay, 1996) (*Singular Value Decomposition - SVD*), proposto em (Liu and Michel, 1994), apresenta uma estratégia de solução de (4) semelhante à do método anterior (Seção 2.1). Considere que a equação (4) seja resolvida a partir de:

$$\begin{aligned}x^p - x^m &= T(y^p - y^m), \\ p &= 1, \dots, (m-1).\end{aligned}\quad (8)$$

de forma que o valor de  $T$  encontrado seja usado para obter  $I$  de

$$I = x^m - T y^m. \quad (9)$$

Essa estratégia de solução permite que se obtenha o vetor  $I$  em (9) em função do vetor  $T$  encontrado com a solução de (8). Reescrevendo (8) na forma matricial, obtém-se

$$X = TY, \quad (10)$$

onde  $Y = [y^1 - y^m, y^2 - y^m, \dots, y^{(m-1)} - y^m] \in \mathbb{R}^{n \times (m-1)}$  e  $X = \alpha Y \in \mathbb{R}^{n \times (m-1)}$ , para algum  $\alpha > 1$  (ver comentário a respeito da escolha de  $X$  na Seção 2.1).

Desprezando-se os elementos nulos da matriz esparsa  $T$  (devido à restrição de conectividade local das células da rede - ver comentário na Seção 2.1), a equação anterior pode ser reescrita como

$$X_i = \tilde{T}_i \tilde{Y}_i, \quad i = 1, \dots, n \quad (11)$$

onde  $\tilde{T}_i \in \mathbb{R}^{1 \times v_i}$  denota a  $i$ -ésima linha de  $T$  com as posições não utilizadas já eliminadas,  $\tilde{Y}_i \in \mathbb{R}^{v_i \times (m-1)}$  denota a  $i$ -ésima linha de  $Y$  considerando-se a eliminação dos elementos nulos desta última, e  $X_i \in \mathbb{R}^{1 \times (m-1)}$  denota a  $i$ -ésima linha de  $X$ , sendo que a variável  $v_i$  denota a quantidade de vizinhos da  $i$ -ésima célula.

Uma solução para (11) pode ser obtida como no método anterior (Seção 2.1), por meio de matrizes pseudo-inversas:

$$\hat{T}_i = X_i (\tilde{Y}_i)^+ \quad (12)$$

Além disso, pode-se demonstrar (ver (Liu and Michel, 1994)) que a adição do seguinte termo, contendo um vetor arbitrário  $w_i$ , garante ser esta também uma solução para (11):

$$\begin{aligned}\bar{T}_i &= X_i (\tilde{Y}_i)^+ + w_i U_{i2}^T, \\ i &= 1, \dots, n\end{aligned}\quad (13)$$

onde o termo  $w_i U_{i2}^T$  compreende uma forma de “correção” aplicada à solução por pseudo-inversas dada por (12). A matriz  $U_{i2}$  em (13) é obtida pela fatoração SVD de  $\tilde{Y}_i$  (ver (Lay, 1996)):

$$\tilde{Y}_i = [U_{i1} \mid U_{i2}] \Sigma_i \begin{bmatrix} V_{i1}^T \\ V_{i2}^T \end{bmatrix},$$

onde  $U_{i1} \in \mathbb{R}^{v_i \times d_i}$ ,  $U_{i2} \in \mathbb{R}^{v_i \times (v_i - d_i)}$ ,  $\Sigma_i \in \mathbb{R}^{v_i \times (m-1)}$ ,  $V_{i1} \in \mathbb{R}^{(m-1) \times d_i}$ ,  $V_{i2} \in \mathbb{R}^{(m-1) \times (m-1 - d_i)}$  e  $d_i = \text{posto}(\tilde{Y}_i)$ .

O método de aprendizado considerado nessa seção é sintetizado pelo Algoritmo 2.2.

## 2.3 Método baseado em aprendizado Hebbiano

Este método garante o ponto de equilíbrio dentro de uma região de saturação. Por meio da estimativa da constante  $E$  que

**Algoritmo 2.2** Algoritmo de treinamento baseado em Decomposição por Valor Singular (SVD)

**Require:**  $m$  padrões de treinamento  $y^p, p = 1, \dots, m$

**Ensure:** parâmetros  $T, I$  da RNC

Construa as matrizes

$$Y = [y^1 - y^m, y^2 - y^m, \dots, y^{(m-1)} - y^m], \quad X = \alpha Y$$

**for**  $i = 1, \dots, n$  **do**

Construa  $\tilde{T}_i$  e  $\tilde{Y}_i$  a partir de  $T$  e  $Y$  {ver instruções no texto}

Calcule a Decomposição por Valor Singular de  $\tilde{Y}_i$ :

$$\tilde{Y}_i = [U_{i1} | U_{i2}] \Sigma_i \begin{bmatrix} \frac{V_{i1}^T}{V_{i2}^T} \end{bmatrix}$$

Calcule

$$\bar{T}_i = X_i (\tilde{Y}_i)^+ + w_i U_{i2}^T$$

Reconstrua a matriz  $T$  a partir de  $\bar{T}_i$  {conforme descrito na Observação 2.1}

Reconstrua o vetor  $I$  como

$$I_i = x^m - T_i y^m$$

**end for**

define essa região (ver abaixo), pode-se estimar a capacidade de recuperação de dados da rede, uma vez que atingida a região, o sistema caminha para o ponto de equilíbrio da mesma.

O método proposto em (Szolgay et al., 1997) propõe um tratamento alternativo à equação de equilíbrio (4). Na verdade, essa equação pode ser reescrita como um sistema de inequações, utilizando-se a definição do conjunto  $C^m(y^p)$  (ver Seção 2.1):

$$x_i^p y_i^p > 1, \quad i = 1, \dots, n \quad e \quad p = 1, \dots, m \quad (14)$$

onde  $x_i^p$ , que denota o  $i$ -ésimo componente do vetor  $x^p$ , é dado por:

$$x_i^p = T_i y^p + I_i, \quad i = 1, \dots, n \quad e \quad p = 1, \dots, m$$

onde  $T_i$  designa a  $i$ -ésima linha da matriz  $T$  e  $I_i$  o  $i$ -ésimo componente de  $I$ .

Considerando-se que os equilíbrios  $x_i^p$  em (14) devem existir dentro da região de saturação (ver discussão na Seção 2.1), esta região pode ser limitada por uma constante  $E$ :

$$(T_i y^p) y_i^p + I_i y_i^p > E > 1, \quad i = 1, \dots, n \quad e \quad p = 1, \dots, m. \quad (15)$$

Seja  $\tilde{T}_i$  o vetor obtido a partir da eliminação dos componentes de  $T_i$  iguais a zero, devido à restrição de conectividade local de RNCs (ver discussão na Seção 2.1). De maneira correspondente, considere o vetor  $\tilde{y}^p$ , obtido a partir da eliminação dos componentes de  $y^p$  que são multiplicados por zeros em  $T_i$ . Então, a inequação (15) resulta em

$$(\tilde{T}_i \tilde{y}^p) y_i^p + I_i y_i^p > E > 1, \quad i = 1, \dots, n \quad e \quad p = 1, \dots, m. \quad (16)$$

Seja

$$\bar{T}_i = [\tilde{T}_i, I_i] \\ \bar{y}^p = [\tilde{y}^p \cdot y_i^p, y_i^p].$$

Da inequação (16), tem-se

$$\bar{T}_i \bar{y}^p > E > 1, \quad i = 1, \dots, n \quad e \quad p = 1, \dots, m. \quad (17)$$

Essas inequações são resolvidas pela regra de plasticidade Hebbiana (Hebb, 1949):

$$\bar{T}_i = \sum_{p=1}^m w_i^p (\bar{y}_i^p)^T, \quad i = 1, \dots, n \quad (18)$$

onde  $w_i^p$  representa um “fator de ganho” (ou taxa de aprendizado). Substituindo (18) em (17) obtém-se as “correlações” entre os padrões. O Algoritmo 2.3 estabelece um procedimento iterativo para o treinamento da rede.

A constante  $E > 1$  é escolhida dentro de um intervalo  $[E_L; E_R]$  com  $E_L > 1$ , sendo que a obtenção de um valor ótimo é investigada experimentalmente em (Szolgay et al., 1997). Nessa referência, também é obtida a seguinte condição de convergência do algoritmo:

$$0 < \lambda < \frac{1}{d \cdot m} \quad (19)$$

onde  $d$  é o número de parâmetros livres (dimensão de  $\bar{T}_i$ ) a serem determinados e  $m$  é o número de padrões a serem armazenados.

---

**Algoritmo 2.3** Algoritmo de treinamento baseado em aprendizado Hebbiano
 

---

**Require:**  $m$  padrões de treinamento  $y^p, p = 1, \dots, m$ 
**Ensure:** parâmetros  $T, I$  da RNC

**for**  $i = 1, \dots, n$  **do**
 $\bar{T}_i(0) = 0$ 
**end for**
**for**  $k = 0, 1, 2, \dots$  **do**
**for**  $i = 1, \dots, n$  e  $p = 1, \dots, m$  **do**
 $E_i^p(k) = \bar{T}_i(k) \cdot \bar{y}_i^p$ 
**if**  $E_i^p(k) \neq [E_L; E_R]$  **then**
 $\delta w_i^p = -\lambda \cdot (E_i^p(k) - E)$  {ver equação (19)}

**else**
 $\delta w_i^p = 0$ 
**end if**

$$\bar{T}_i(k+1) = \bar{T}_i(k) + \sum_{p=1}^m \delta w_i^p \cdot (\bar{y}_i^p)^T$$
**end for**

 Parar quando  $\delta w_i^p = 0$ , para todo  $i = 1, \dots, n$  e para todo  $p = 1, \dots, m$ .

**end for**


---

## 2.4 Método baseado em Perceptrons

O método baseado em Perceptron possui a vantagem adicional de não precisar impor restrições na diagonal da matriz  $T$  para garantir a existência de pontos de equilíbrio conforme é necessário, por exemplo, para redes de Hopfield ( $T_{ii} = 0$ ).

O método apresentado em (Liu and Lu, 1997) utiliza Perceptrons (Rosenblatt, 1958) para resolver o sistema de inequações dados por (14). Este sistema pode ser reescrito como

$$\begin{cases} T_i y_i^p + I_i > 1 & \text{se } y_i^p = 1 \\ T_i y_i^p + I_i < -1 & \text{se } y_i^p = -1 \end{cases} \quad (20)$$

para  $i = 1, \dots, n$  e  $p = 1, \dots, m$ , onde  $T_i$  representa a  $i$ -ésima linha de  $T$ ,  $I_i$  o  $i$ -ésimo elemento de  $I$  e  $y_i^p$  o  $i$ -ésimo elemento de  $y^p$ .

Defina

$$\begin{aligned} W^i &= [w_1^i, w_2^i, \dots, w_i^i + \mu_i, \dots, w_n^i, I_i] \\ \bar{y}^p &= [y^p, 1] \end{aligned}$$

onde  $w_j^i = T_{ij}$ , caso a  $j$ -ésima célula pertença à vizinhança da  $i$ -ésima célula e  $w_j^i = 0$ , caso contrário. A constante  $\mu_i > 1$  foi introduzida para que a inequação (20) possa ser reescrita como:

$$\begin{cases} W^i \bar{y}^p \geq 0, & \text{se } y_i^p = 1 \\ W^i \bar{y}^p < 0, & \text{se } y_i^p = -1 \end{cases} \quad (21)$$

o que garante a semelhança desta última inequação com a definição de um Perceptron (Rosenblatt, 1958):

$$z = \begin{cases} 1, & \text{se } Gu \geq 0 \\ -1, & \text{se } Gu < 0. \end{cases} \quad (22)$$

onde  $G \in \mathbb{R}^{1 \times n}$  representa os pesos da rede (a serem encontrados),  $u \in \mathbb{R}^n$  representa um padrão de entrada, e  $z = \pm 1$  a classe a que o padrão de entrada pertence.

Dessa forma, os parâmetros  $W^i$  podem ser obtidos de (21) aplicando-se o algoritmo de aprendizado do Perceptron (para maiores detalhes desse algoritmo, refira-se a (Liu and Lu, 1997)). O procedimento de treinamento proposto nesta seção é sintetizado pelo Algoritmo 2.4.

---

**Algoritmo 2.4** Algoritmo de treinamento baseado em aprendizado por Perceptron
 

---

**Require:**  $m$  padrões de treinamento  $y^p, p = 1, \dots, m$ 
**Ensure:** parâmetros  $T, I$  da RNC

**for**  $i = 1, \dots, n$  **do**

Treinar o Perceptron

$$W^i = [w_1^i, w_2^i, \dots, w_i^i + \mu_i, \dots, w_n^i, I_i]$$

 para que satisfaça ( $p = 1, 2, \dots, m$ ):

$$\begin{cases} W^i \bar{y}^p \geq 0, & \text{se } y_i^p = 1 \\ W^i \bar{y}^p < 0, & \text{se } y_i^p = -1 \end{cases} \quad (23)$$

**for**  $j = 1, 2, \dots, n$  **do**
**if**  $j$  pertence a vizinhança de  $i$  **then**
**if**  $i \neq j$  **then**
 $T_{ij} = w_j^i$ 
**else**
 $T_{ii} = w_i^i + \mu_i$ 
**end if**
**else**
 $T_{ij} = 0$ 
**end if**
**end for**
**end for**


---

## 2.5 Método baseado em LMIs

As redes de Hopfield requerem que  $T$  seja simétrica. Utilizando o método baseado em LMIs, descrito na seqüência para o projeto de RNCs, além da vantagem em relação a não dependência do estado inicial da rede, pode-se trabalhar com

uma matriz  $T$  assimétrica. Com isso, o projeto de memórias associativas torna-se mais flexível.

Em (Park et al., 2001) é proposto um sistema de inequações equivalente a (14) através da introdução das variáveis  $\kappa_i$  que controlam o tamanho das bacias de atração em torno dos padrões a serem armazenados, conforme o teorema que segue.

**Teorema 2.2** Dado um padrão  $y^* \in B^n = \{-1, 1\}^n$ , suponha que para  $i = 1, \dots, n$ , os parâmetros  $T$  e  $I$  de (2) satisfaçam

$$y_i^* \left( \sum_{j \in Z_i} T_{ij} y_j^* + I_i \right) > \kappa_i \max_{j \in Z_i} |T_{ij}| + (T_{ii} - 1) \quad (24)$$

onde  $\kappa_i \geq 0$  e onde  $Z_i$  denota o conjunto contendo os índices das células vizinhas à  $i$ -ésima célula. Então, um vetor binário  $y^0 \in B^n$  tal que  $y_i^0 \neq y_i^*$  e  $\sum_{j \in Z_i} |y_j^0 - y_j^*| < \kappa_i$ , tem as seguintes propriedades:

- (a) se  $x(0) = y^0$  em (2), então  $x_i(t)$  tende a  $y_i^*$  em  $t = 0$ ;
- (b) o vetor  $y^0$  não corresponde a nenhum equilíbrio da equação (2).

**Prova:** ver referência (Park et al., 2001).

Este teorema, junto com o corolário apresentado a seguir, são os principais resultados utilizados na derivação da regra de aprendizado para a RNC desta seção.

**Corolário 2.3** Suponha  $T = [T_{ij}]$  simétrica em (2) e

$$T_{ii} > 1, \quad i = 1, \dots, n. \quad (25)$$

Então, todo ponto de equilíbrio [assintoticamente estável]  $x^* = [x_1^*, \dots, x_n^*]^T$  de (2) satisfaz a condição

$$|x_i^*| > 1, \quad i = 1, \dots, n. \quad (26)$$

**Prova:** ver referência (Liu and Michel, 1994).

O corolário anterior propõe uma condição em  $T$  para que os padrões armazenados sejam binários (bipolares), isto é, ele propõe uma condição que garante que os pontos de equilíbrios estejam dentro da região de saturação (isto é,  $|x_i^*| > 1$ ).

**Observação 2.4** A inequação (24) é equivalente a:

$$\begin{aligned} y_i(T_i y^p + I_i) &= T_{ii} + y_i \left( \sum_{j \in Z_i} T_{ij} y_j + I_i \right) \\ &> \kappa_i \max_{j \in Z_i} |T_{ij}| + (2T_{ii} - 1) \end{aligned}$$

e, considerando-se  $\kappa_i \geq 0$  e  $T$  sob a hipótese de (25), da inequação anterior tem-se:

$$(T_i y^p + I_i) y_i > 1.$$

Portanto, o critério dado por (24) no Teorema 2.2 estabelece uma condição equivalente a (14) para que os padrões  $y^p$  ( $p = 1, \dots, m$ ) sejam armazenados na rede.

Dessa forma, os autores do método propõem como solução ao problema de memória associativa a solução de um problema de otimização associado, dado por (24), e formulado como um GEVP (*Generalized Eigenvalue Problem* ou Problema de Autovalor Generalizado) (Park et al., 2001).

Observa-se que maximizando  $\kappa_i$  em (24), quando esse problema é factível, implica que as bacias de atração em torno dos padrões são maximizadas, como desejado. Isto implica, por sua vez, minimizar  $T_{ii}$  em (24), que só pode assumir o valor mínimo de  $T_{ii} = 1$ , de acordo com a hipótese de (25). Assim, para  $T_{ii} = 1$  em (24), esse problema é factível se

$$y_i \left( \sum_{j \in Z_i} T_{ij} y_j + I_i \right) > 0, \quad i = 1, \dots, n \quad (27)$$

for factível. Os parâmetros da RNC devem ser obtidos maximizando  $\kappa_i \geq 0$  em (24), com  $T_{ii} = 1$ , tal que:

$$y_i \left( \sum_{j \in Z_i} T_{ij} y_j + I_i \right) > \kappa_i \max_{j \in Z_i} |T_{ij}|, \quad i = 1, \dots, n \quad (28)$$

Considerando-se que os parâmetros da RNC são limitados ( $|T_{ij}| < U$  e  $|I_i| < U$ , para  $i = 1, \dots, n$  e  $U > 0$ ), a não-linearidade da inequação anterior pode ser eliminada. Para isso, considere o caso não-trivial em que  $\max_{j \in Z_i} |T_{ij}| \neq 0$  e  $\kappa_i > 0$ . Introduzindo-se a variável limitada  $q_i \in (L, U)$ , onde  $L > 0$  e  $U > 0$ , pode-se reescrever (28) como:

$$y_i \left( \sum_{j \in Z_i} T_{ij} y_j + I_i \right) > \kappa_i \max_{j \in Z_i} |T_{ij}| > \kappa_i q_i, \quad i = 1, \dots, n$$

A inequação anterior pode ser separada em

$$y_i \left( \sum_{j \in Z_i} T_{ij} y_j + I_i \right) - \kappa_i q_i > 0, \quad i = 1, \dots, n$$

e

$$|T_{ij}| < q_i, \quad j \in Z_i.$$

Note-se que essas são Inequações Matriciais Lineares (*Linear Matrix Inequalities - LMIs*) (Park et al., 2001), onde cada variável  $T_{ij}$ ,  $I_i$ ,  $\kappa_i$  e  $q_i$  é uma “matriz” de ordem  $1 \times 1$ . Então, quando (27) for factível, o problema a ser resolvido consiste em maximizar  $\kappa_i$ , para  $T_{ii} = 1$ , a partir da inequação anterior:

$$\begin{aligned} & \min (-\kappa_i) \\ & \text{s. a.} \begin{cases} (-\kappa_i)q_i + y_i \left( \sum_{j \in Z_i} T_{ij} y_j + I_i \right) > 0, \\ |T_{ij}| < q_i, \quad j \in Z_i \\ |I_i| < U \\ L < q_i < U, \end{cases} \\ & i = 1, \dots, n. \end{aligned}$$

Caso este problema não seja factível, a solução deve ser obtida exclusivamente de (14) e (25):

$$\begin{cases} (T_{ii} y^p + I_i) y_i = T_{ii} + y_i \left( \sum_{j \in Z_i} T_{ij} y_j + I_i \right) > 1 \\ T_{ii} \geq 1, \quad i = 1, \dots, n. \end{cases}$$

Para valores elevados de  $T_{ii}$ , essa condição é trivialmente satisfeita mas, no entanto, o número de estados espúrios (equilíbrios indesejados) é aumentado. Assim, procura-se fixar  $T_{ii} = 1 + \epsilon$ , obtendo-se uma solução não-trivial que minimize  $\epsilon$ , de acordo com

$$y_i \left( \sum_{j \in Z_i} T_{ij} y_j + I_i \right) > -\epsilon, \quad i = 1, \dots, n.$$

O Algoritmo 2.5 resume o procedimento proposto para treinamento da RNC.

---

### Algoritmo 2.5 Algoritmo de treinamento baseado em LMIs

---

**Require:**  $m$  padrões de treinamento  $y^p$ ,  $p = 1, \dots, m$

**Ensure:** parâmetros  $T, I$  da RNC

**for**  $i = 1, \dots, n$  **do**

**if**  $y_i^p \left( \sum_{j \in Z_i} T_{ij} y_j^p + I_i \right) > 0$  é factível ( $p = 1, \dots, m$ )

**then**

Fixe  $T_{ii} = 1$  e resolva o GEVP dado por

$$\begin{aligned} & \min (-\kappa_i) \\ & \text{s. a.} \begin{cases} -\kappa_i q_i + y_i^p \left( \sum_{j \in Z_i} T_{ij} y_j^p + I_i \right) > 0, \\ -q_i < T_{ij} < q_i, \quad j \in Z_i \\ -U < I_i < U, \\ L < q_i < U \end{cases} \\ & p = 1, \dots, m \end{aligned}$$

**else**

Fixe  $T_{ii} = 1 + \epsilon$  e encontre os parâmetros restantes resolvendo

$$\begin{cases} y_i^p \left( \sum_{j \in Z_i} T_{ij} y_j^p + I_i \right) > -\epsilon, \\ -q_i < T_{ij} < q_i, \quad j \in Z_i \\ -U < I_i < U. \end{cases} \\ k = 1, \dots, m$$

**end if**  
**end for**

---

## 3 EXPERIMENTOS

Nos experimentos realizados, procurou-se avaliar o desempenho dos métodos apresentados nas seções anteriores quanto à capacidade de armazenamento e de recuperação de padrões consistindo de letras do alfabeto (A-Z), conforme ilustrado na Figura 2 (imagens obtidas de (Park et al., 2001)). Cada subdivisão nas figuras corresponde a um pixel da imagem, que é associado a uma célula da rede. Portanto, as RNCs simuladas tiveram dimensões de  $7 \times 7$  ( $n = 49$ ) células. Nas imagens consideradas, a cor branca corresponde ao valor -1 e a cor preta ao valor 1, com valores intermediários representados em uma escala de tons de cinza.

Os experimentos avaliaram o comportamento dos métodos de treinamento sob às seguintes variações de parâmetros:

- raio de vizinhança ( $r$ ): (ver equação (1)) corresponde à quantidade de vizinhos em torno de cada célula da rede. Foi variado em:  $r = 1, 2, 3$ ;



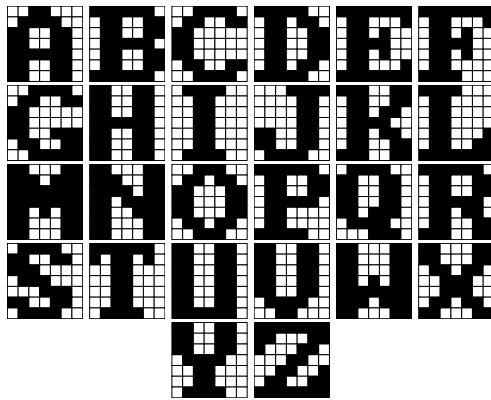


Figura 2: Padrões utilizados para o treinamento da rede.

- taxa de carga: corresponde à razão da quantidade de padrões armazenados pelo tamanho da rede (quantidade de células). Foi variada em: 10%, 20%, 30%, 40% e 50% (que corresponderam ao armazenamento de  $p = 5, 10, 15, 20, 25$  padrões, respectivamente);
- níveis de ruído: correspondem ao grau em que os padrões de teste foram corrompidos por ruído. Foram estudados dois tipos de ruído:
  - binário, cujos níveis utilizados foram: 2%, 4%, 6%, 10%, 15% e 20%, que correspondem à porcentagem de pixels (ou bits) invertidos nos padrões de treinamento;
  - Gaussiano, cujos níveis estudados foram: 0.3, 0.5, 0.7, 1.0, 1.5 e 2.0, que correspondem a diferentes desvios padrões para a distribuição normal de média zero utilizada na geração de tal ruído.

Para cada combinação de parâmetros, e para cada método de treinamento, foi medida a taxa média de recuperação dos padrões, que corresponde à média de sucessos na recuperação dos padrões obtida para um determinado número de tentativas.

No texto que segue, considere a seguinte legenda: *P-Inversa* denota o método da Seção 2.1; *Perceptron* indica o método da Seção 2.4; *SVD* corresponde ao método da Seção 2.2; *Hebb* denota o método da Seção 2.3 e *LMI* corresponde ao método da Seção 2.5.

Parâmetros específicos de cada método foram também estudados e os testes apresentam resultados obtidos com a escolha do valor do parâmetro que gerou o melhor desempenho, dentro de uma faixa de valores investigada. Os valores sublinhados a seguir, correspondem aos valores de parâmetros que foram utilizados para a análise comparativa dos métodos:

- *P-Inversa*:  $\alpha = 2, 3, 5, \underline{7}, 10$ ;
- *Perceptron*:  $\mu = \underline{1.2}, 2.0, 3.0, 5.0, 7.0$ ;
- *SVD*:  $\alpha = 2, 3, 5, \underline{7}, 10$ ,  $w_i = \mu_i \cdot O_{m_i} \cdot U_{i2}$ , onde  $\mu_i = -10, -5, \underline{1}, 5, 10$  e  $O = [1, \dots, 1] \in R^{1 \times v_i}$ ;
- *Hebb*:  $[E_L, E_R] = [\underline{1.1}, 3.1], [2.0, 4.0], [4.0, 6.0], [1.1, 10.0], [1.1, 21.1]$ ,  $E = (E_L + E_R)/2$ ;
- *LMI*:  $\epsilon = 0.0001, 0.001, 0.01, \underline{0.1}, 0.2$ ,  $(L, U) = (1, 3), (2, 4), (4, 6), (\underline{1}, 10), (1, 20)$ .

Foram considerados 10 conjuntos de dados de treinamento para cada diferente taxa de carga (sendo 5 taxas de cargas consideradas, conforme mencionado acima). Cada conjunto consistiu de 30 amostras de padrões aleatoriamente selecionados dentre os padrões apresentados na Figura 2. Esses conjuntos de dados foram usados para treinar as RNCs pelos diferentes métodos apresentados na seção anterior. Além disso, na avaliação das RNCs, é também realizado um comparação com o desempenho de redes de Hopfield para os mesmo conjuntos de dados.

Para testar a capacidade de recuperação de padrões pelos métodos, os padrões provenientes dos 10 conjuntos de treinamento foram perturbados com ruído (ruído binário e Gaussiano, em 6 diferentes níveis) e as taxas médias de recuperação foram medidas. Os padrões foram considerados corretamente recuperados quando 100% dos pixels foram restaurados nas imagens, durante 50 passos de tempo<sup>1</sup> em que as equações foram integradas. O algoritmo utilizado para integração numérica da equação (2) foi o Runge-Kutta de quarta ordem (Press et al., 1993). Os experimentos foram implementados com auxílio do software MATLAB (The MathWorks, 2000) em um computador com processador Intel Core Duo 1.66GHz com 1 GB de RAM. Os problemas de otimização no Algoritmo 2.5 foram resolvidos por métodos de ponto-interior (Boyd et al., 1994), implementados pelas funções *feasp* e *gevp* do *toolkit* LMI.

## 4 RESULTADOS

Os gráficos apresentados nas Figuras 3 e 4 (ver Página 461) resumem os resultados obtidos com os experimentos. Observa-se que os métodos LMI e Hebb se mostraram ligeiramente superiores nos testes envolvendo ruído binário. Com relação aos testes envolvendo ruído Gaussiano, ambos os métodos se mostraram equivalentes, com exceção ao método LMI, que para o parâmetro de vizinhança  $r = 1$  teve um desempenho muito inferior.

Deve-se destacar que o desempenho reportado para o método *Perceptron* foi conseguido com a introdução de uma

<sup>1</sup>O tempo foi discretizado em incrementos  $\Delta t = 0.2$

modificação no algoritmo, uma vez que o desempenho deste se mostrou extremamente insatisfatório, comparativamente com o dos demais métodos. A condição para o armazenamento dos padrões, estabelecida pela inequação (23) do Algoritmo 2.4, foi modificada para:

$$\begin{cases} W^i \bar{y}^p \geq 1, & \text{se } y_i^p = 1 \\ W^i \bar{y}^p < 1, & \text{se } y_i^p = -1 \end{cases} \quad (29)$$

$i = 1, \dots, n$  e  $p = 1, \dots, m.$

Dessa forma, os pontos de equilíbrio foram ajustados mais no interior da região de saturação, aumentando assim a robustez do método na recuperação dos padrões.

Para se determinar quais foram os métodos que obtiveram melhor desempenho nos experimentos, foi empregado o teste estatístico denominado “Teste T de Student” (Morettin and de O. Bussab, 2003), utilizando-se um nível de significância de 5%. As Tabelas 1 e 2 relacionam os melhores métodos encontrados, para cada variação de parâmetros considerada nos experimentos.

Tabela 1: Melhores métodos - recuperação de padrões corrompidos por ruído *binário*.

Carga	r = 1	r = 2	r = 3
10%	P-Inv/SVD/Hebb/LMI	Hebb/LMI	Hebb/LMI
20%	Hebb/LMI	Hebb/LMI	Hebb/LMI
30%	Hebb/LMI	Hebb/LMI	Hebb/LMI
40%	Percept/Hebb/LMI	Hebb/LMI	Hebb/LMI
50%	Hebb/LMI	Hebb/LMI	Hebb/LMI

Tabela 2: Melhores métodos - recuperação de padrões corrompidos por ruído *Gaussiano*.

Carga	r = 1	r = 2	r = 3
10%	Todos equiv.	Todos equiv.	Percept/Hebb/LMI
20%	Todos equiv.	Todos equiv.	Percept/Hebb/LMI
30%	Todos equiv.	Todos equiv.	Percept/Hebb/LMI
40%	Todos equiv.	Todos equiv.	Percept/Hebb/LMI
50%	Todos equiv.	Todos equiv.	Percept/Hebb/LMI

Os gráficos apresentados nas Figuras 5 (ver Página 462) ilustram o desempenho da rede de Hopfield para os mesmo conjuntos de dados utilizados para avaliar as RNCs. Pode-se observar que essas redes têm sua taxa de recuperação de padrões reduzida significativamente com o aumento da carga. Por outro lado, o comportamento das RNCs Hebb e LMI é claramente melhor utilizando  $r = 3$ . Este resultado revela um diferencial dessas RNCs para armazenamento de grande quantidade de dados.

Outro aspecto relevante em relação ao desempenho das técnicas investigadas é o custo computacional de cada uma. A Ta-

bela 3 apresenta os tempos de computação requeridos pelas RNCs. Comparando o tempo de computação da rede de Hopfield, 0,3750 segundos, com os tempos da Tabela 3, verifica-se que a rede de Hopfield requer tempo menor que o das RNCs Hebb e LMI. Porém, deve-se observar que as RNCs podem ser efetivamente paralelizadas em hardware, tornando o tempo de processamento dos métodos Hebb e LMI apropriados para aplicações em tempo real.

Tabela 3: Tempos de computação (em segundos) para as RNCs investigadas variando o raio de vizinhança.

Métodos	r = 1	r = 2	r = 3
P-Inv	0.0320	0.0620	0.1250
Perceptron	0.0620	0.0780	0.1090
SVD	0.3280	0.5940	0.9530
Hebb	6.9380	13.3440	24.0470
LMI	20.4690	66.2810	156.7350

## 5 CONCLUSÃO

Apresentou-se, neste artigo, uma análise comparativa de desempenho de diversos métodos de construção de memória associativa em RNCs. Além disso, procurou-se fornecer uma revisão desses métodos, como propostos na literatura. A análise realizada forneceu condições para que os métodos estudados pudessem ser comparados de maneira homogênea e criteriosa. Os resultados obtidos encontram-se em conformação com resultados parciais anteriores, obtidos na literatura. Espera-se que este estudo forneça condições que auxiliem na escolha do melhor método de construção de memória associativa em RNCs para o desenvolvimento de aplicações práticas. Também espera-se que os resultados da análise possam servir de guia na busca de aperfeiçoamentos dos métodos estudados, bem como de outros, a serem desenvolvidos.

## REFERÊNCIAS

- Boyd, S., ElGhaoui, L., Feron, E. and Balakrishnan, V. (1994). *Linear matrix inequalities in systems and control theory*, SIAM, Philadelphia, PA.
- Chua, L. O. and Yang, L. (1988). Cellular neural networks: theory, *IEEE Transactions on Systems and Circuits* **35**(10): 1257–1272.
- Crounse, K. R. and Chua, L. O. (1995). Methods for image processing and pattern formation in cellular neural networks: a tutorial, *IEEE Transactions on Circuits and Systems* **42**(10): 583–601.
- Grassi, G. (2001). On discrete-time cellular neural networks for associative memories, *IEEE Transactions on Circuits and Systems* **48**(1): 107–111.

- Hebb, D. O. (1949). *The organization of behavior*, Wiley, New York.
- Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities, *Proc. of the National Academy of Sciences of the USA* **79**: 2554–2558.
- Israel, A. B. and Greenville, T. N. E. (1974). *Generalized inverses: theory and applications*, Wiley, New York.
- Kaneko, K. (1989a). Pattern dynamics in spatiotemporal chaos, *Physica D* **34**: 1–41.
- Kaneko, K. (1989b). Spatiotemporal chaos in one- and two-dimensional coupled map lattices, *Physica D* **37**: 60–82.
- Kozek, T., Chua, L. O., Roska, T., Wolf, D., Tetzlaff, R., Puffer, F. and Lotz, K. (1995). Simulating nonlinear waves and pdes via cnn - part ii: typical examples, *IEEE Transactions on Circuits and Systems* **42**(10): 816–820.
- Lay, D. C. (1996). *Linear algebra and its applications*, 2nd edn, Addison-Wesley.
- Liu, D. and Lu, Z. (1997). A new synthesis approach for feedback neural networks based on the perceptron training algorithm, *IEEE Transactions on Neural Networks* **8**(6): 1468–1482.
- Liu, D. and Michel, A. N. (1994). Sparsely interconnected neural networks for associative memories with applications to cellular neural networks, *IEEE Transactions on Circuits and Systems* **41**: 295–307.
- Michel, A. N. and Farrell, J. A. (1990). Associative memories via artificial neural networks, *IEEE Control Systems Magazine* pp. 6–17.
- Morettin, P. A. and de O. Bussab, W. (2003). *Estatística básica*, 5 edn, Saraiva, São Paulo.
- Park, J., Kim, H. Y. and Lee, S. W. (2001). A synthesis procedure for associative memories based on space-varying cellular neural networks, *Neural Networks* (14): 107–113.
- Press, W. H., Flannery, B. P., Teukolsky, S. A. and Vetterling, W. T. (1993). *Numerical recipes in C: the art of scientific computing*, Cambridge University Press.
- Rosenblatt, F. (1958). The perceptron: probabilistic model for information storage and organization in the brain, *Psychological Review* (65): 386–408.
- Roska, T., Chua, L. O., Wolf, D., Kozek, T., Tetzlaff, R. and Puffer, F. (1995). Simulating nonlinear waves and pdes via cnn - part i: basic techniques, *IEEE Transactions on Circuits and Systems* **42**(10): 807–815.
- Szolgay, P., Szatmári, I. and László, K. (1997). A fast fixed point learning method to implement associative memory on cnns, *IEEE Transactions on Circuits and Systems* **44**(4): 362–366.
- The MathWorks, I. (2000). Matlab, CD-ROM. Versão 6, Release 12.
- Thiran, P., Crouse, K. R., Chua, L. O. and Hasler, M. (1995). Pattern formation properties of autonomous cellular neural networks, *IEEE Transactions on Circuits and Systems CAS* **42**: 757–776.
- Werblin, F., Roska, T. and Chua, L. (1995). The analogic cellular neural network as a bionic eye, *Int. J. Circuit Theory and Applications* **23**: 541–569.
- Wolfram, S. (1983). Statistical mechanics of cellular automata, *Rev. Mod. Phys.* **55**: 601–644.
- Wolfram, S. (1986). *Theory and application of cellular automata*, World Scientific, Singapore.

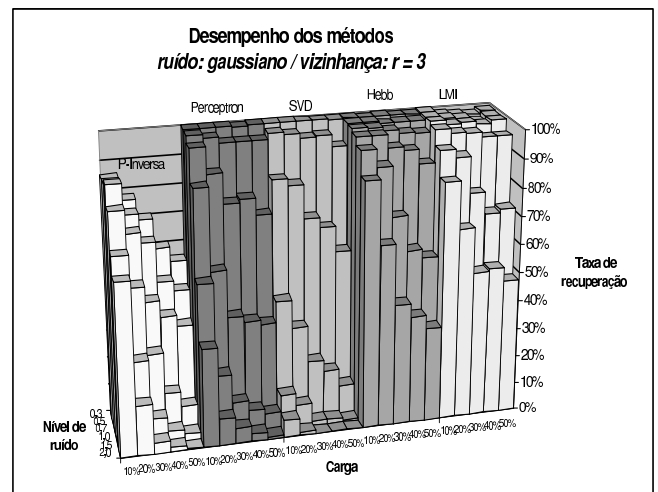
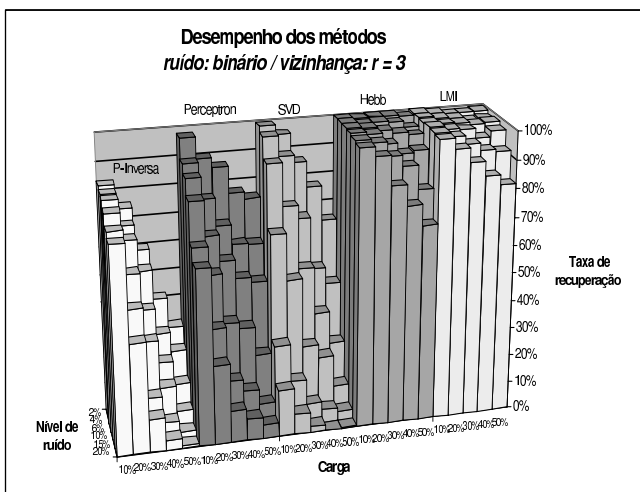
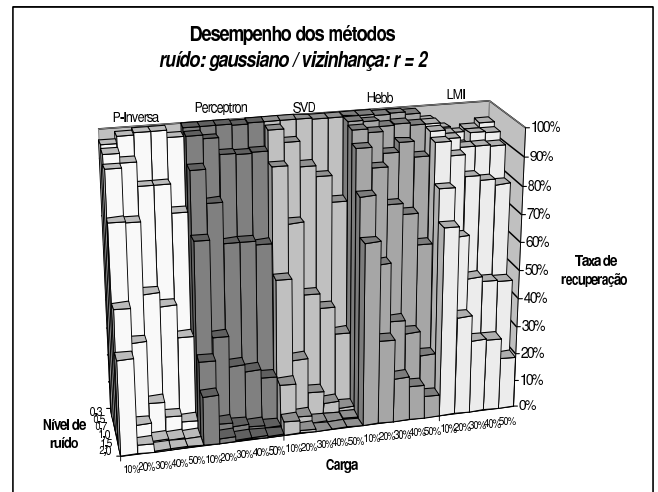
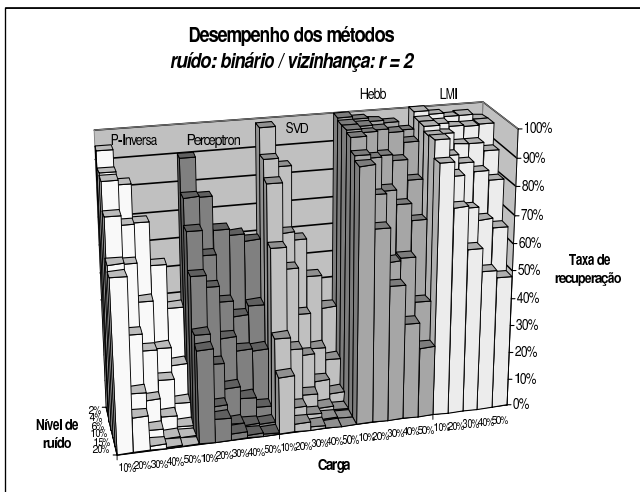
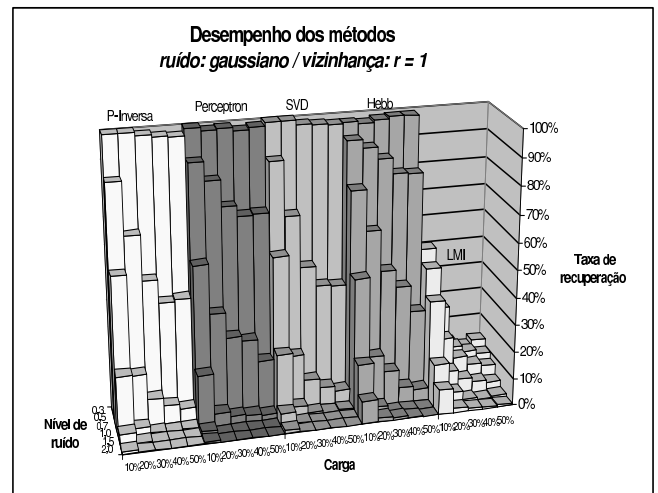
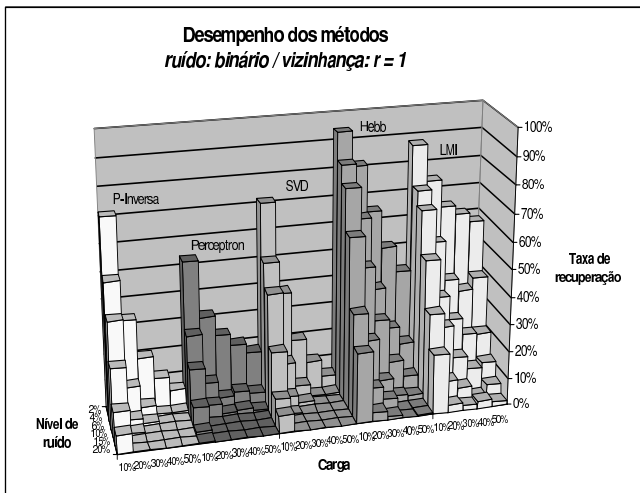


Figura 3: Gráficos de desempenho para ruído binário.

Figura 4: Gráficos de desempenho para ruído Gaussiano.

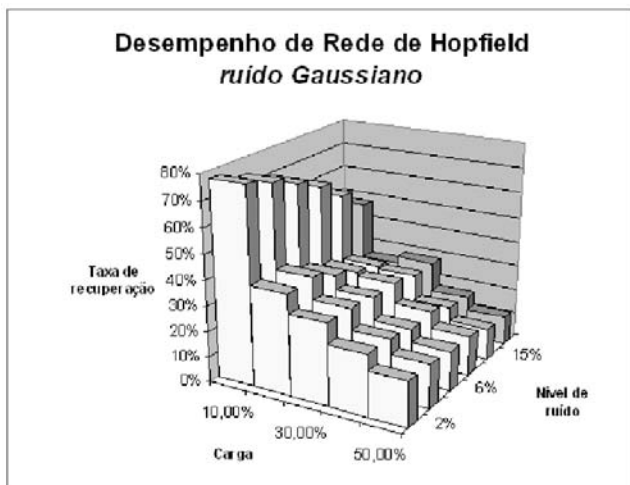
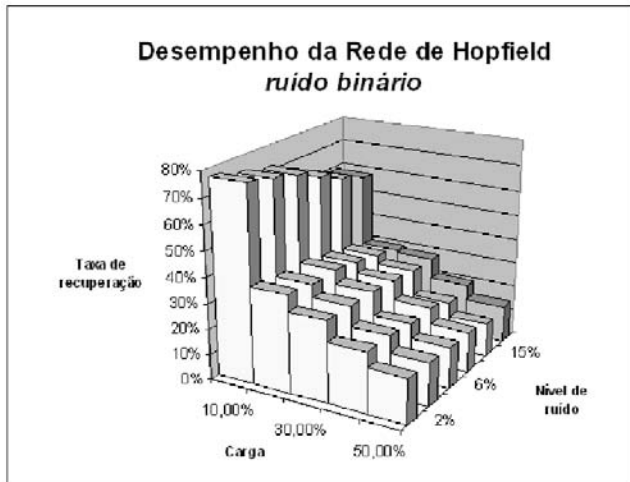


Figura 5: Gráficos de desempenho da rede de Hopfield para ruídos binário e Gaussiano.