UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

GUSTAVO BRUNO DO VALE

RECONSTRUÇÃO E RECONHECIMENTO DE IMAGENS BINÁRIAS UTILIZANDO O ALGORITMO MÁQUINA DE *BOLTZMANN*

Uberlândia
2016

GUSTAVO BRUNO DO VALE

RECONSTRUÇÃO E RECONHECIMENTO DE IMAGENS BINÁRIAS UTILIZANDO O ALGORITMO MÁQUINA DE *BOLTZMANN*

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, da Universidade Federal de Uberlândia (MG), como parte dos requisitos para a obtenção do título de Mestre em Ciências.

Área de Concentração: Processamento da Informação.

Orientador: Prof. Dr. Gilberto Arantes Carrijo

Banca examinadora:

Prof. Dr. Gilberto Arantes Carrijo, UFU
Prof. Dr. Antônio Cláudio Paschoarelli Veiga, UFU
Prof. Dr. Wilson Tavares Ferreira, IFMT
Profª Drª Milena Bueno Pereira Carneiro, UFU

Uberlândia

2016

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

V149r
2016

Vale, Gustavo Bruno do, 1986-
Reconstrução e reconhecimento de imagens binárias utilizando o algoritmo Máquina de Boltzmann / Gustavo Bruno do Vale. - 2016.
107 f. : il.

Orientador: Gilberto Arantes Carrijo.
Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Engenharia Elétrica.
Inclui bibliografia.

1. Engenharia elétrica - Teses. 2. Reconhecimento de padrões - Teses. 3. Redes neurais (Computação) - Teses. 4. Algoritmos - Teses. I. Carrijo, Gilberto Arantes. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Engenharia Elétrica. III. Título.

CDU: 621.3

GUSTAVO BRUNO DO VALE

RECONSTRUÇÃO E RECONHECIMENTO DE IMAGENS BINÁRIAS UTILIZANDO O ALGORITMO MÁQUINA DE *BOLTZMANN*

Dissertação aprovada para a obtenção do título de Mestre em Ciências no Programa de Pós-Graduação em Engenharia Elétrica, da Universidade Federal de Uberlândia (MG).

Uberlândia, 19 de maio de 2016.

_____

Prof. Dr. Gilberto Arantes Carrijo, UFU/MG
Orientador

_____

Prof. Dr. Darizon Alves de Andrade, UFU/MG
Coordenador do Curso de Pós-Graduação

Dedico este trabalho à minha esposa Celene,
aos meus pais Valdeci e Vilma e a todos meus
familiares e amigos que me acompanham e
torcem por mim.

AGRADECIMENTOS

Primeiramente, agradeço a Deus por ter me abençoado e me ajudado a concluir mais esta etapa da vida.

À minha família e amigos, em especial à minha esposa Celene Cristina Dias Silva, por sempre estar ao meu lado me ajudando e incentivando em todas as minhas decisões.

Aos meus pais, Valdeci Ferreira do Vale e Vilma Rosa Bruno Ferreira, pelo apoio que sempre me deram aos meus estudos.

Em especial, expresso meus sinceros agradecimentos ao meu professor orientador Dr. Gilberto Arantes Carrijo por todo incentivo, colaboração, confiança e constantes ensinamentos ao longo deste trabalho.

À professora Dra. Maria Teresa Menezes Freitas, diretora do Centro de Educação a Distância da UFU (CEAD), pelo apoio e incentivo dado à qualificação dos servidores do CEAD.

Ao Programa de Pós-Graduação em Engenharia Elétrica da UFU, pela oportunidade de realização de trabalhos em minha área de pesquisa.

"Todo aquele que se dedica ao estudo da ciência chega a convencer-se de que nas leis do Universo se manifesta um Espírito sumamente superior ao do homem, e perante o qual nós, com os nossos poderes limitados, devemos humilhar-nos."

Albert Einstein

RESUMO

O objetivo deste trabalho é utilizar algoritmos conhecidos como Máquina de *Boltzmann* para reconstruir e classificar padrões como de imagens. Este algoritmo possui uma estrutura parecida com a de uma Rede Neural Artificial porém os nós da rede possuem decisões estocásticas e probabilísticas. Neste trabalho é apresentado o referencial teórico das principais Redes Neurais Artificiais, do algoritmo Máquina de *Boltzmann* Geral e uma variação deste algoritmo conhecida como Máquina de *Boltzmann* Restrita. São realizadas simulações computacionais comparando os algoritmos Rede Neural Artificial *Backpropagation* com estes algoritmos Máquina de *Boltzmann* Geral e Máquina de *Boltzmann* Restrita. Através de simulações computacionais são analisados os tempos de execuções dos diferentes algoritmos descritos e os percentuais de acerto de *bits* dos padrões treinados que são posteriormente reconstruídos. Por fim, são utilizadas imagens binárias com e sem ruído no treinamento do algoritmo Máquina de *Boltzmann* Restrita, estas imagens são reconstruídas e classificadas de acordo com o percentual de acerto de *bits* na reconstrução das imagens. Os algoritmos Máquina de *Boltzmann* foram capazes de classificar padrões treinados e apresentaram ótimos resultados na reconstrução dos padrões com um rápido tempo de execução do código podendo assim ser utilizado em aplicações como de reconhecimento de imagens.

Palavras-chaves: Máquina de Boltzmann, Reconhecimento de imagens, Reconhecimento de padrões, Redes Neurais Artificiais.

ABSTRACT

The objective of this work is to use algorithms known as Boltzmann Machine to rebuild and classify patterns as images. This algorithm has a similar structure to that of an Artificial Neural Network but network nodes have stochastic and probabilistic decisions. This work presents the theoretical framework of the main Artificial Neural Networks, General Boltzmann Machine algorithm and a variation of this algorithm known as Restricted Boltzmann Machine. Computer simulations are performed comparing algorithms Artificial Neural Network Backpropagation with these algorithms Boltzmann General Machine and Machine Restricted Boltzmann. Through computer simulations are analyzed executions times of the different described algorithms and bit hit percentage of trained patterns that are later reconstructed. Finally, they used binary images with and without noise in training Restricted Boltzmann Machine algorithm, these images are reconstructed and classified according to the bit hit percentage in the reconstruction of the images. The Boltzmann machine algorithms were able to classify patterns trained and showed excellent results in the reconstruction of the standards code faster runtime and thus can be used in applications such as image recognition.

Keywords: Boltzmann Machine, Recognition of images, Pattern Recognition, Artificial Neural Networks.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BM *Boltzmann Machine*

RBM *Restricted Boltzman Machine*

RNA Rede Neural Artificial

CAPÍTULO 1

1. INTRODUÇÃO

Sistemas de reconhecimento de padrões são processos muito utilizados atualmente para se realizar o reconhecimento de faces, identificação de assinaturas, autenticação de usuários, identificação através da íris do olho, reconhecimento de impressões digitais, reconhecimento de fala e várias outras finalidades. Tais sistemas tem o objetivo de facilitar, agilizar e aumentar a segurança deste tipo de processo. Para se realizar o reconhecimento de tais padrões, estes sistemas analisam os diferentes padrões de entrada descritos e posteriormente são realizados os reconhecimentos e a classificações destes dados.

Para solucionar os problemas descritos no parágrafo anterior, são utilizadas diferentes técnicas de reconhecimento de padrões e algumas delas utilizam um algoritmo conhecido como Rede Neural Artificial (RNA). Além disso, pesquisas têm sido desenvolvidas sobre a utilização de tipos especiais de RNAs que utilizam dados estocásticos com decisões probabilísticas para treinamento de padrões a serem aprendidos. Neste trabalho utiliza-se os algoritmos Máquina de *Boltzmann* Geral e Máquina de *Boltzmann* Restrita que possuem tais características estocásticas e probabilísticas. Compara-se a acurácia destes algoritmos com o algoritmo RNA *Backpropagation* que é mais conhecido e utilizado.

As pesquisas realizadas destes algoritmos são recentes e poucos trabalhos foram realizados comparando estes algoritmos. Os resultados a serem alcançados neste trabalho são importantes pois apresentam a viabilidade de se utilizar estas ferramentas computacionais em aplicações que necessitam de reconhecimento e classificação de padrões.

Além disso, esta pesquisa aplica o algoritmo Máquina de *Boltzmann* Restrita na reconstrução e classificação de imagens binárias. Através de simulações computacionais, faz-se uma análise da carga computacional desta aplicação e, principalmente, compara-se o percentual de reconhecimento de *bits* destas imagens binárias.

1.1. RECONHECIMENTO DE IMAGENS

Auxiliados por avanços na capacidade de computação e tecnologia de processamento de imagens, aplicações voltadas para reconhecimento de imagens têm sido bastante utilizadas atualmente. Além disso, sistemas de reconhecimento especializados estão se tornando mais

susceptíveis a satisfazer as rigorosas restrições na precisão e velocidade, bem como no custo do desenvolvimento e da manutenção.

Um sistema de reconhecimento de imagens completo normalmente consiste de vários componentes como: pré-processamento, detecção, segmentação, extração de características e classificação. Vários métodos e algoritmos são utilizados para reconhecer e classificar imagens, como: Redes Neurais Artificiais, Algoritmos Genéticos, métodos de Distância Euclidiana, Análise de Componentes Principais, entre outros.

Neste trabalho, utiliza-se algoritmos ainda pouco explorados para reconhecimento de imagens. Compara-se também o algoritmo RNA *Backpropagation* com os algoritmos Máquina de *Boltzmann* Geral e Máquina de *Boltzmann* Restrita para classificação de padrões. Por fim, é simulada a reconstrução e classificação de imagens binárias com o algoritmo Máquina de *Boltzmann* Restrita.

1.2. METODOLOGIA DE PESQUISA UTILIZADA

A metodologia adotada neste trabalho foi uma pesquisa aplicada a reconstrução, reconhecimento e classificação de imagens. Nesta pesquisa realizada, buscou-se identificar ferramentas computacionais capazes de solucionar problemas de reconhecimento de padrões. Através de uma pesquisa bibliográfica identificou-se estudos feitos por pesquisadores sobre RNAs e algoritmos Máquina de *Boltzmann*.

Com tais informações, iniciou-se a fase de pesquisa experimental comparando os resultados obtidos de aplicações utilizando diferentes algoritmos de reconhecimento, classificação e reconstrução de padrões. Ao final do trabalho, foram utilizadas imagens binárias que passaram por um processo de reconstrução e reconhecimento.

1.3. OBJETIVOS

Esta pesquisa desenvolvida teve por objetivo, testar os algoritmos Máquina de Boltzmann Geral e Máquina de Boltzmann Restrita na reconstrução de imagens binárias. Além disso, foi possível utilizar tais algoritmos na classificação de padrões baseados nos percentuais de acertos de bits dos diferentes algoritmos simulados. Os resultados obtidos dos algoritmos testados foram comparados às simulações do algoritmo RNA *Backpropagation*.

CAPÍTULO 2

2. REDES NEURAIS ARTIFICIAIS

2.1. INTRODUÇÃO

As redes neurais artificiais são sistemas de processamento de informações inspirados nas redes neurais biológicas. Estas redes neurais apresentam modelos matemáticos de um neurônio artificial que visam aprender padrões baseados em treinamentos para posteriormente realizar decisões [1].

Em uma rede neural várias unidades chamadas neurônios são interconectados mantendo pesos de ligação e esse conjunto forma uma rede. Através dessa rede as unidades de entrada são processadas no neurônio e a saída é determinada por esse processamento e por uma função de ativação. Essa rede é capaz de aprender padrões baseado no treinamento no qual é submetida [1]. As redes neurais são utilizadas na resolução de problemas que envolvem reconhecimento de padrões, predição, otimização, memória associativa e controle [2].

2.2. HISTÓRICO

Os primeiros trabalhos realizados na área de Redes Neurais e no desenvolvimento do modelo do neurônio artificial foi fruto do trabalho de Warren McCulloch e Walter Pitts em 1943. McCulloch era psiquiatra e neuroanatomista e Pitts era um matemático recém-formado. McCulloch dedicou 20 anos de estudos na representação de um evento no sistema nervoso e Pitts juntou-se ao seu trabalho em 1942. O trabalho de ambos era na busca de descrever um modelo artificial de um neurônio e apresentar suas capacidades computacionais [3].

Estudos relacionados ao aprendizado de redes biológicas e artificias foram realizadas apenas alguns anos após o trabalho de McCulloch e Pitts. Donald Hebb em 1949 realizou o primeiro trabalho que se tem notícia relacionado com este aprendizado, Hebb mostra que o aprendizado de redes neurais é conseguido através da variação dos pesos de entradas dos nós da rede. Ele propôs uma teoria para explicar o aprendizado dos nós biológicos através do reforço das ligações sinápticas entre os nós excitados, esta regra é conhecida em algoritmos de aprendizagem com regra de *Hebb*. Posteriormente, Widrow e Hoff sugeriram uma outra regra

conhecida como regra delta, baseado em um método do gradiente para minimização do erro na saída de um neurônio [3].

Em 1962, Frank Rosenblatt criou um modelo conhecido como Modelo *Perceptron* de uma única camada que foi capaz de provar a convergência de um algoritmo de aprendizado. Este modelo era capaz de classificar padrões aprendidos através do ajuste gradual dos pesos da rede. Porém Marvin Minsky e Seymour Papert [4] provaram incapacidade deste modelo em solucionar problemas não linearmente separáveis como "Ou Exclusivo". Por isso, a comunidade de ciência da computação deixou o paradigma de redes neurais por mais de 20 anos.

Após alguns anos, novos modelos mais eficientes para redes multiníveis surgiram, provocando novamente forte interesse por essa área. Pode-se citar o modelo de *Hopfield* [5] [6], das Máquinas de *Boltzmann* [7] e do algoritmo *Backpropagation* [8], que contrariaram um pouco as idéias de Minsky e Papert sobre *Perceptrons* e provocaram novamente um forte interesse pela área com publicações de outros modelos importantes de Redes Neurais [2].

Acreditava-se existir uma memória no sistema e construiu-se uma teoria estatística, utilizando a ideia de mecanismos estatísticos, incorporando a ideia de aprendizado do cérebro proposta por Hebb. Hopfield [5] utilizou esta ideia introduzindo uma energia funcional, enfatizando a noção de memória como atratores dinamicamente estáveis. Em 1985 Hilton e Sejnowski [7] construíram formulações usando unidades estocásticas [2].

2.3. MOTIVAÇÃO DAS RNAS: NEURÔNIOS BIOLÓGICOS

No cérebro humano existem cerca de 10^{11} neurônios, cada um destes neurônios se comunica entre si de forma paralela e continua. O cérebro é formado por uma rede de neurônios com conexões dos nós e o comportamento em conjunto desta rede são objetos de estudos das RNAs [3].

O cérebro humano é responsável pela emoção, pensamento e cognição, também executa funções sensoriais, motoras e autônomas. Além disso, essa rede de neurônios tem a capacidade de reconhecer padrões, usar e armazenar conhecimento por experiência, e interpretar observações. Por isso, as RNAs tentam reproduzir as funções dessas redes biológicas, buscando implementar seu comportamento e sua dinâmica [3].

Os neurônios são divididos em três partes: o corpo celular, os dendritos e o axônio. Os dendritos têm a função de receber informações por impulsos nervosos, oriundas de outros neurônios e conduzi-las até o corpo celular. No corpo celular, a informação é processada, e novos impulsos são gerados. Estes impulsos são transmitidos a outros neurônios, passando através do axônio até os dendritos dos neurônios seguintes. O ponto de contato entre a terminação axônica de um neurônio e o dendrito de outro é chamado de sinapse. É através dessas sinapses que os nós se unem, formando redes neurais, além disso, elas são capazes de controlar a transmissão de impulsos, ou o fluxo de informações entre os nós da rede neural. A figura 2.1 a seguir mostra de forma simples os principais componentes do neurônio. O neurônio é responsável pela maioria das funções realizadas por nosso cérebro. A capacidade de nosso cérebro realizar funções complexas, deve-se ao fato da execução em paralelo dos 10^{11} nós do nosso cérebro [3].

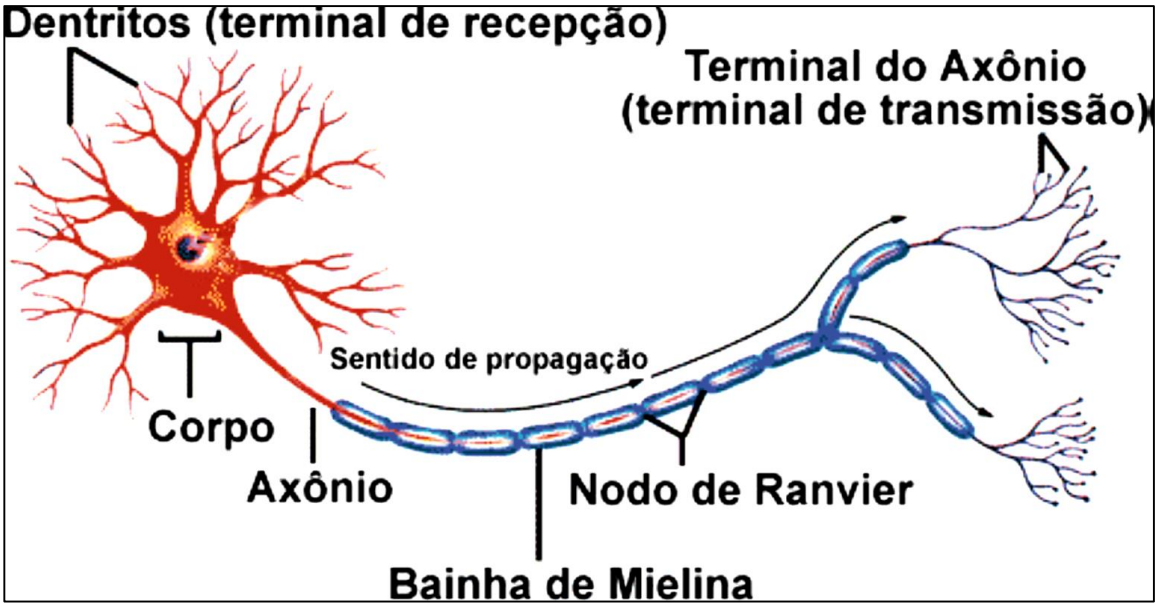

Figura 2.1 - A célula nervosa (Neurônio) [2].

A comunicação do cérebro pode ser feita através de sinais químicos pelas sinapses e através de sinais elétricos dentro do neurônio. A membrana que envolve a parte externa do corpo do neurônio tem a capacidade de gerar impulsos nervosos (elétricos). O corpo, por sua vez, combina os sinais recebidos e, se o valor resultante for acima do limiar de excitação do neurônio, um impulso elétrico é produzido e propagado através do axônio para os neurônios seguintes [3].

Ao receber um estímulo, a membrana do neurônio se despolariza, ocasionando a inversão das cargas elétricas do interior e exterior do neurônio [2]. Isto faz com que o impulso nervoso se propague pelo axônio até suas conexões sinápticas. Quando este impulso chega nos terminais do axônio, moléculas neurotransmissoras são liberadas para dentro da clave sináptica e o processo continua no neurônio seguinte. O tipo de neurotransmissor liberado determinará a polarização ou a despolarização do corpo do neurônio seguinte. De acordo, com o tipo de neurotransmissor liberado, a sinapse poderá ser inibitória ou excitatória. Esta sinapse pode ser ajustada através de atividades que ela participa, desta forma a rede neural pode aprender padrões [3]. Devido a alta conectividade desta rede, este sistema torna-se totalmente robusto.

2.4. NEURÔNIO ARTIFICIAL: MODELO MCCULLOC PITTS

Em 1943, McCulloc e Pitts [9] desenvolveram um modelo computacional, inspirados na estrutura e propriedades fisiológicas dos neurônios biológicos e de suas conexões. Este modelo de um neurônio artificial é apresentado na figura 2.2.

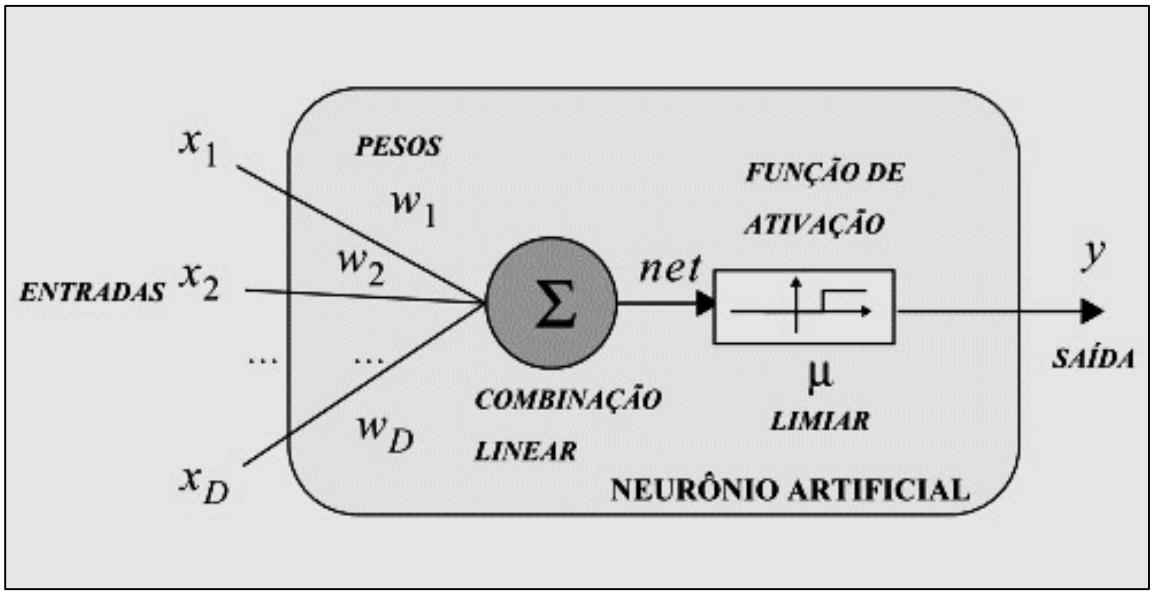

Figura 2.2 - Modelo do neurônio de McCulloch e Pitts [10].

Este neurônio de *McCulloc-Pitts* (MCP) foi um dispositivo desenvolvido que possui várias entradas (x_1 x_2 ... x_D) com um estado, possui pesos (w_1 w_2 ... w_D) nas conexões e uma saída representada por dois estados possíveis (1 pulso e 0 sem pulso) [2]. O processamento que ocorre no neurônio consiste de um somatório dos produtos das entradas e pesos $x_i w_i$

representado na expressão: $net = x_1w_1 + x_2w_2 + ... + x_Dw_D$. Se resultado deste somatório "*net*" ultrapassar um limiar μ, o neurônio dispara o valor 1 na saída y, se não ultrapassar este limiar a saída fica passiva em y = 0.

A função de ativação do modelo *McCulloc-Pitts*, não é a única função que pode ser utilizada para definir a saída do neurônio. Existem outras funções de ativação que são utilizadas nas RNAs para determinar a saída y do neurônio artificial. Na figura 2.3 é apresentada algumas funções de ativação: a função linear que apresenta uma saída linear contínua, a função de escada que possui uma saída binária (não-linear discreta) e a função sigmoidal.

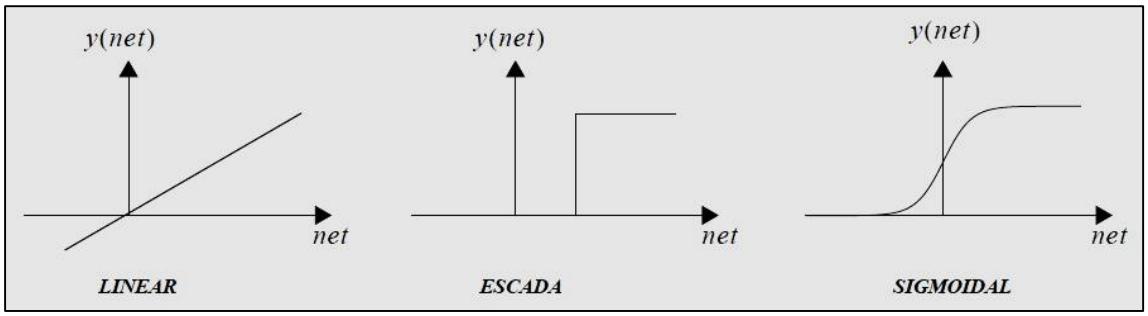

Figura 2.3 - Funções de ativação [10].

Apesar da capacidade de resolver alguns problemas, o neurônio de McCulloc-Pitts apresenta algumas limitações. Redes MCP de apenas uma camada só consegue implementar funções linearmente separáveis, além disso, o modelo foi proposto com pesos fixos, não-ajustáveis [3].

2.5. ARQUITETURAS DAS RNAS

As RNAs podem apresentar diferentes arquiteturas que dependerá da forma que os neurônios são conectados e as redes são formadas. Redes com uma única camada de nós como de *McCulloc-Pitts*, só conseguem resolver problemas linearmente separáveis. Redes recorrentes, por outro lado, são mais apropriadas para solucionar problemas que envolvem processamento temporal. Para se definir a arquitetura da rede, leva-se em conta o número de camadas da rede e o tipo de conexão entre os nós. Na figura 2.4 são apresentadas as principais arquiteturas das RNAs, na Figura 2.4 (a) apresenta uma rede de uma camada única e na Figura 2.4 (b) apresenta uma rede de múltipla camada, no qual existe mais de um neurônio entre a camada de entrada e camada de saída da rede.

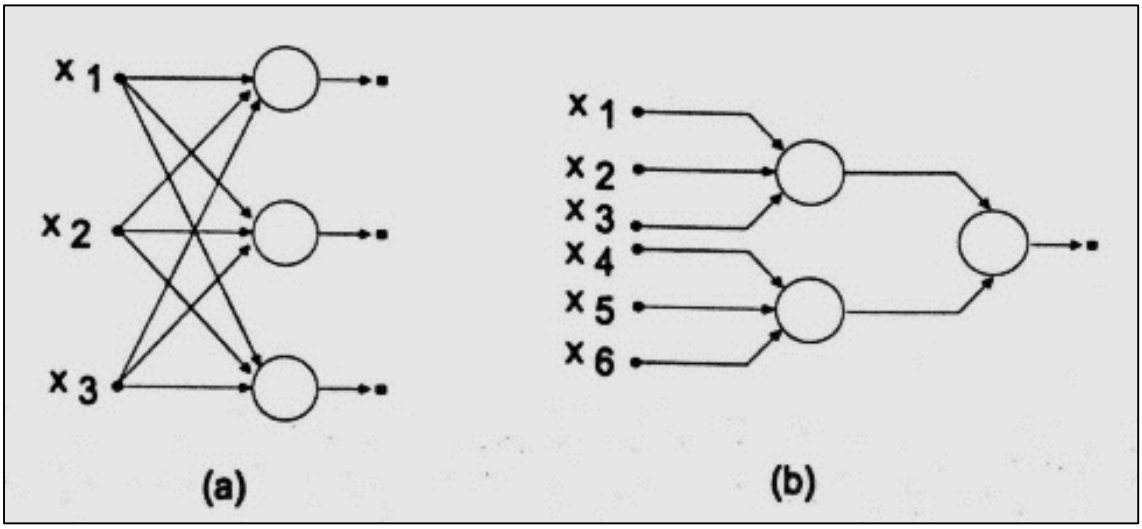

Figura 2.4 - Arquiteturas de RNAs [3].

2.6. APRENDIZAGEM DE HEBB

Donald Hebb, foi um psicólogo da Universidade de McGill que projetou o primeiro aprendizado para RNA em 1949 [1]. Sua premissa foi que se dois neurônios fossem ativados simultaneamente, então a força de conexão entre eles deveria ser incrementada; porém se dois neurônios são ativados assincronamente então a força daquela sinapse é enfraquecida ou eliminada. Para este modelo a regra de aprendizagem se traduz na equação 2.1 a seguir, onde $\boldsymbol{\eta}$ é a taxa de aprendizagem que determina a velocidade da aprendizagem e $\boldsymbol{y}$ é o valor do neurônio.

$$\text{Regra de aprendizagem de } \textit{Hebb}\text{: } \Delta \boldsymbol{w_{ij}} = \boldsymbol{\eta y_i y_j} \tag{2.1}$$

De acordo com esta regra de *Hebb*, a mudança de pesos sinápticos depende apenas das atividades dos dois neurônios *i* e *j* conectados.

2.7. PERCEPTRONS

Juntos com outros pesquisadores, Frank Rosenblatt introduziu e desenvolveu uma grande classe de RNAs chamadas *Perceptrons*. O mais típico *Perceptron* consiste de uma camada de entrada conectada por caminhos com pesos fixados ao neurônio que está associado; e os pesos dessas conexões são ajustáveis. A regra de aprendizagem *Perceptron* usa um ajuste interativo de pesos que é mais poderoso do que a regra de *Hebb*. Da mesma forma que os

neurônios desenvolvidos por McCulloch e Pitss e por Hebb, *Perceptrons* também usa um limiar na função de saída.

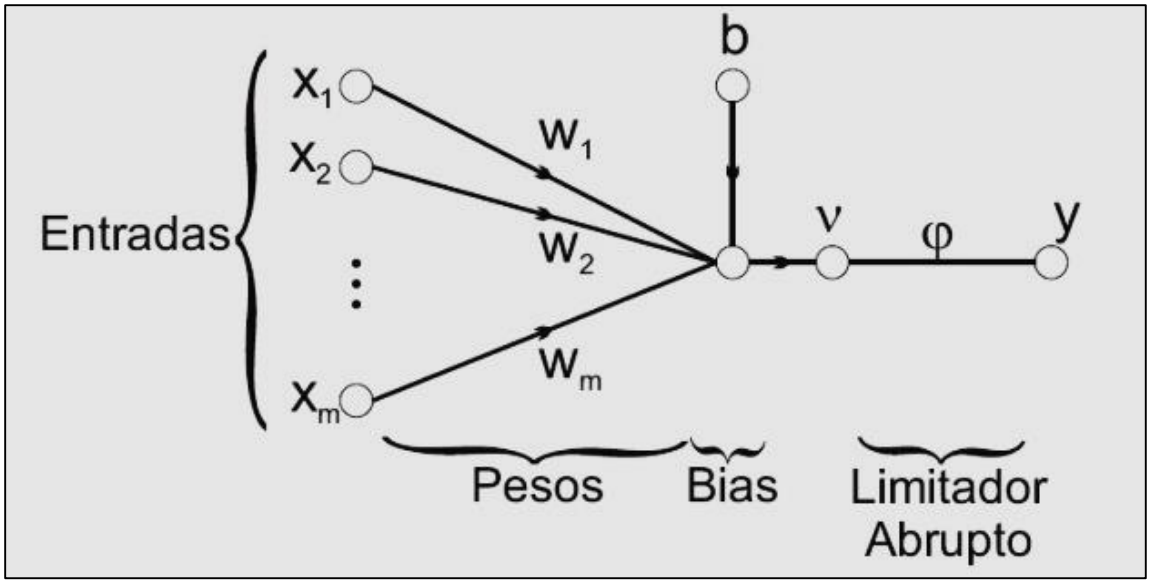

Figura 2.5 - Fluxo de sinal do *Perceptron* [11].

A figura 2.5 apresenta o fluxo de sinais no *Perceptron*. Cada entrada é ponderada por um peso sináptico, sendo x_i na entrada *i* conectado ao neurônio *j* e multiplicado pelo seu peso sináptico w_{ij}. Este modelo inclui, um *bias "b"*, que é uma entrada fixa com o efeito de aumentar ou diminuir a entrada líquida da função ativação. O neurônio realiza a soma de todas essas as entradas ponderadas por seus respectivos pesos sinápticos juntamente como o *bias "b"*. O resultado desta combinação é definido pela equação 2.2:

$$v = \sum_{i=1}^{m} w_i x_i + b \tag{2.2}$$

Este valor é aplicado a um limitador abrupto que produz uma saída igual a +1 se a combinação for positiva e -1 se a combinação for negativa, conforme equação 2.3.

$$y = \begin{cases} 1, se\ v > 0 \\ -1, se\ v < 0 \end{cases} \tag{2.3}$$

Nas redes *Perceptrons*, utiliza-se um treinamento da rede para ajustar o valor dos pesos. Neste treinamento compara-se o valor de saída esperado *t* com o valor *y* encontrado. Caso ocorra um erro na saída *y* da rede em relação ao valor esperado ***d*** pelo treinamento da rede, este

algoritmo realiza a adaptação dos pesos conforme equação 2.4, onde o valor alvo $\boldsymbol{t}$ poderá ser +1 ou -1 e $\boldsymbol{\eta}$ é a taxa de aprendizagem.

$$w(n + 1) = w(n) + \eta t x(n) \qquad (2.4)$$

Após esta atualização do vetor de pesos, novamente é realizado o somatório da equação 2.2 e encontrada a nova saída *y*. Se novamente ocorrer erro na comparação da saída esperada e saída encontrada, faz-se uma nova atualização do vetor de pesos.

Através do *Perceptron* é possível classificar corretamente os estímulos de entradas, desde que as classes sejam linearmente separáveis.

2.8. ADALINE

Bernard Widrow e seu estudante, Marcian Edward "Ted" Hoff, desenvolveram uma regra de aprendizagem, normalmente chamada de regra dos mínimos dos quadrados médios ou regra delta, que está bem relacionada com a regra de aprendizagem *Perceptron*. Este nome ADALINE é interpretado como Neurônio Adaptativo Linear (*Adaptive Linear Neuron*) ou Sistema Adaptativo Linear (*Adaptive Linear System*).

A regra *Perceptron* ajusta os pesos das conexões de uma unidade quando a resposta da unidade está incorreta. Porém, a regra delta ajusta os pesos para reduzir a diferença entre a unidade de saída da rede e a saída desejada. Isto resulta em um menor erro médio quadrático. A regra de aprendizagem *Widrow-Hoff* para uma rede de camada simples é uma precursora da regra *Backpropagation* para redes multicamadas que será apresentada no tópico 2.10 deste trabalho [1].

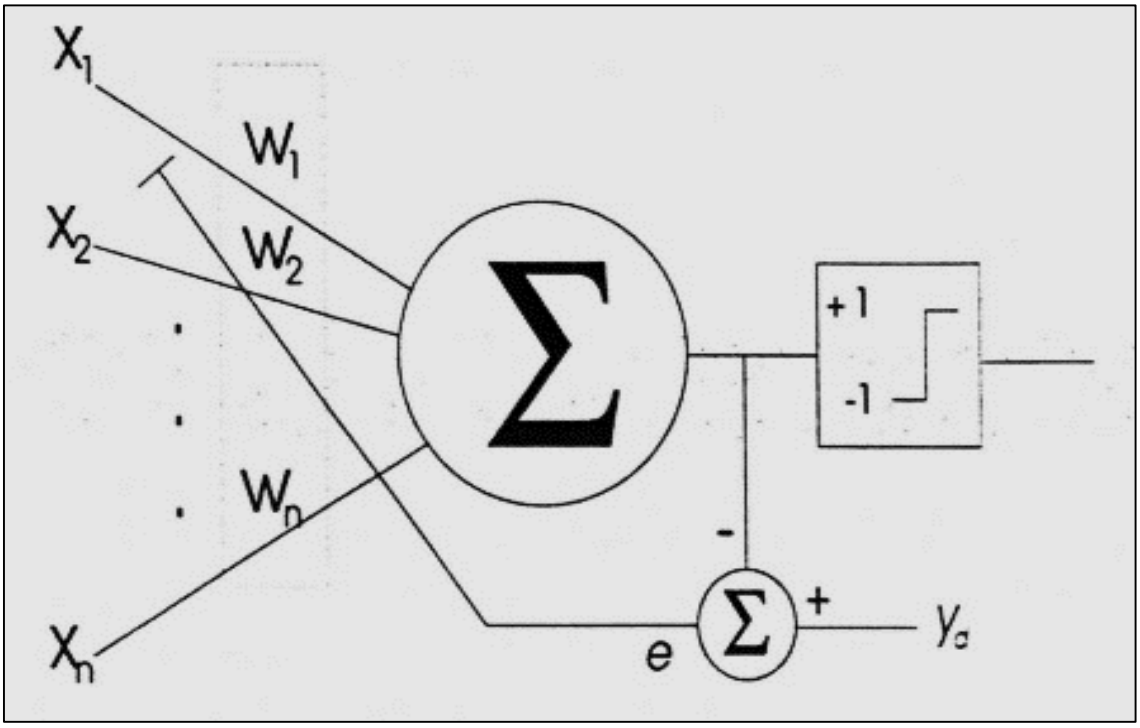

Figura 2.6 - Diagrama esquemático de um nó *Adaline* [3].

A figura 2.6 ilustra o diagrama esquemático de um nó *Adaline*. Neste modelo *Adaline*, permite-se que as entradas e saídas sejam contínuas, as saídas também são determinadas pela função de ativação, porém os pesos são ajustados em função do erro da saída, antes da aplicação da função de ativação. O algoritmo de treinamento *Adaline* minimiza o erro das saídas em relação aos valores desejados t_i pertencentes ao conjunto de treinamento. Para isto, a função erro a ser minimizada é a soma dos erros quadráticos descritos na Equação 2.5:

$$E = \frac{1}{2}\sum_{i=1}^{p}(t^i - v^i)^2 \tag{2.5}$$

Também pode-se deduzir que Δw_i é variação dos pesos proporcional ao negativo da derivada do erro com relação a cada peso, conforme expresso na Equação 2.6.

$$\Delta w_i = \alpha(d^P - y^P)x_i^P \tag{2.6}$$

Onde:

t - valor esperado;

y - valor encontrado;

x - valor de entrada da rede; e

p - padrão ***p*** escolhido.

Após a atualização dos pesos, o algoritmo executa novamente o somatório das entradas ponderadas e os pesos são atualizados novamente de forma a reduzir o erro. Este processo será repetido até encontrar o critério de parada de acordo com a tolerância de erro aceitável ou de acordo com o número máximo de execuções permitidas.

2.9. REDES *PERCEPTRON* MULTICAMADAS

Durante os anos de 1970 os estudos de RNAs ficaram parados devido às limitações dos *Perceptrons* de camada simples em resolver problemas como a função *Xor* e devido à ausência de um método de treinamento de redes multicamadas. Um método para propagar informações sobre os erros nas unidades de saída de volta para as unidades ocultas foi descoberto por Werbos em 1974, mas não ganhou ampla divulgação. Este método também foi descoberto de forma independente por David Parker em 1985 e por LeCun em 1986 antes dele se tornar amplamente conhecido [1].

Para solucionar problemas não linearmente separáveis foi necessário utilizar redes com uma ou mais camadas intermediárias ou ocultas. Segundo Cybenko [12], uma rede com uma camada intermediária pode implementar qualquer função contínua, e a utilização de duas camadas intermediárias permite a aproximação de qualquer função [13]. Estas redes RNAs são do tipo *Perceptron* multicamadas, ou MLP (*multilayer perceptron*), que são RNAs que possuem pelo menos uma camada intermediária ou oculta.

O treinamento proposto em uma rede com mais de uma camada baseia-se em gradiente descendente. Para utilizar este método, a função de ativação mais adequada nas redes com múltiplas camadas, é a função sigmoidal. O principal algoritmo utilizado para treinar tais redes, é um algoritmo chamado de *Backpropagation*. Na figura 2.7 é apresentada uma típica rede MLP.

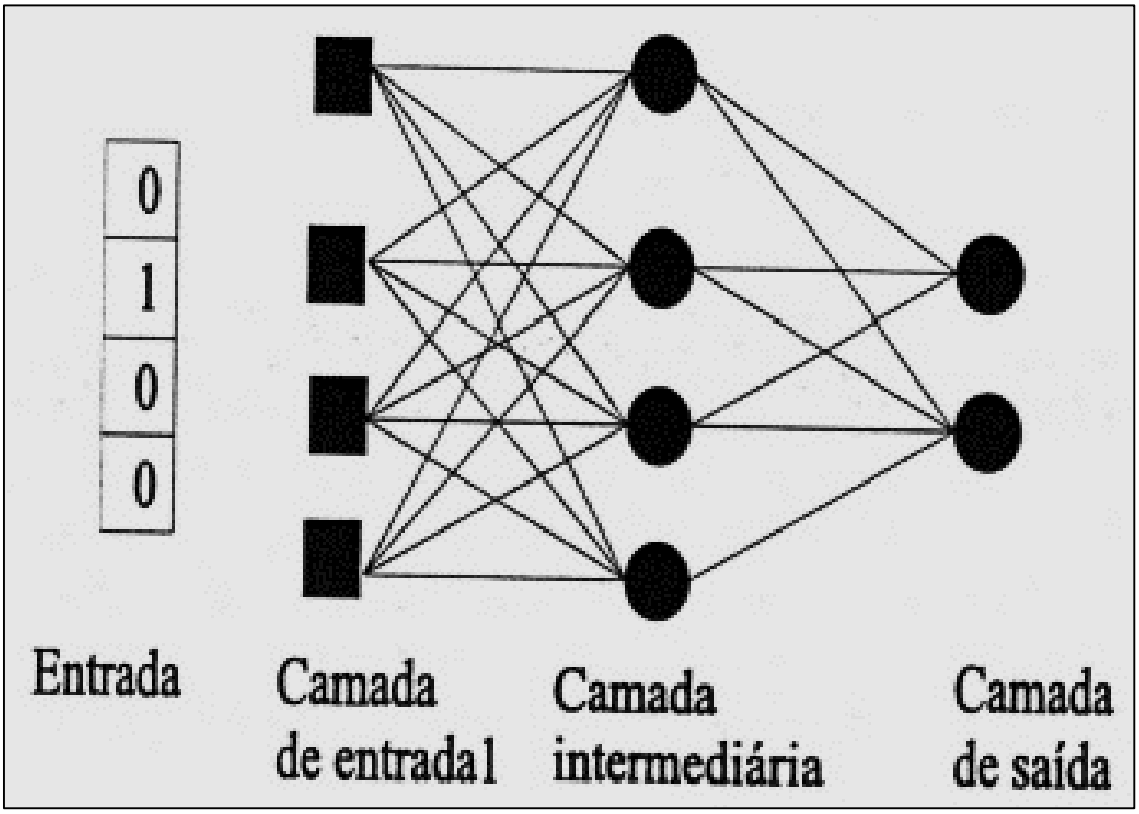

Figura 2.7 - Típica rede MLP com apenas uma camada intermediária [3].

A precisão obtida e a implementação da função objetivo dependem do número de nós utilizados na camada intermediária desta rede. Além disso, no projeto das MLP é importante se preocupar com a função de ativação escolhida, pois a função precisa ser diferenciável para que o ajuste dos pesos possa ser calculado, direcionando assim o ajuste dos pesos.

Para escolha do número de camadas ocultas, deve ser observado que em alguns casos a utilização de duas ou mais camadas intermediárias pode facilitar o treinamento da rede. Já, a utilização de um grande número de camadas intermediárias não é recomendada, pois, cada vez que o erro medido durante a fase de treinamento é propagado para a camada anterior, ele se torna menos preciso.

Com relação ao número de nós nas camadas intermediárias, este deve ser definido empiricamente. Dependerá fortemente da distribuição de padrões de treinamento e validação da rede. Além disso, existem problemas que necessitam apenas de uma unidade de entrada e de uma unidade de saída, e outros que podem necessitar de milhares de unidades intermediárias.

2.10. BACK-PROPAGATION: TREINAMENTO DAS MLPS

Conforme já mencionado, o algoritmo de treinamento mais utilizado nas redes MLPs é o algoritmo B*ackpropagation*. A maioria dos algoritmos de aprendizagem das redes multicamadas utilizam como base este algoritmo B*ackpropagation* e este foi principal responsável pelo retorno do interesse de pesquisadores nos estudos das RNAs.

O algoritmo B*ackpropagation* é um algoritmo supervisionado que utiliza pares (entrada, saída desejada), para realizar a correção de erros através do ajuste dos pesos da rede. O treinamento aconteceu em duas fases, fase *forward* e fase *backward*. A fase *forward* é utilizada para definir a saída de rede de acordo com o padrão entrada. A fase *backward* utiliza a saída desejada e saída encontrada pela rede para atualizar os pesos das conexões, conforme apresenta a figura 2.8.

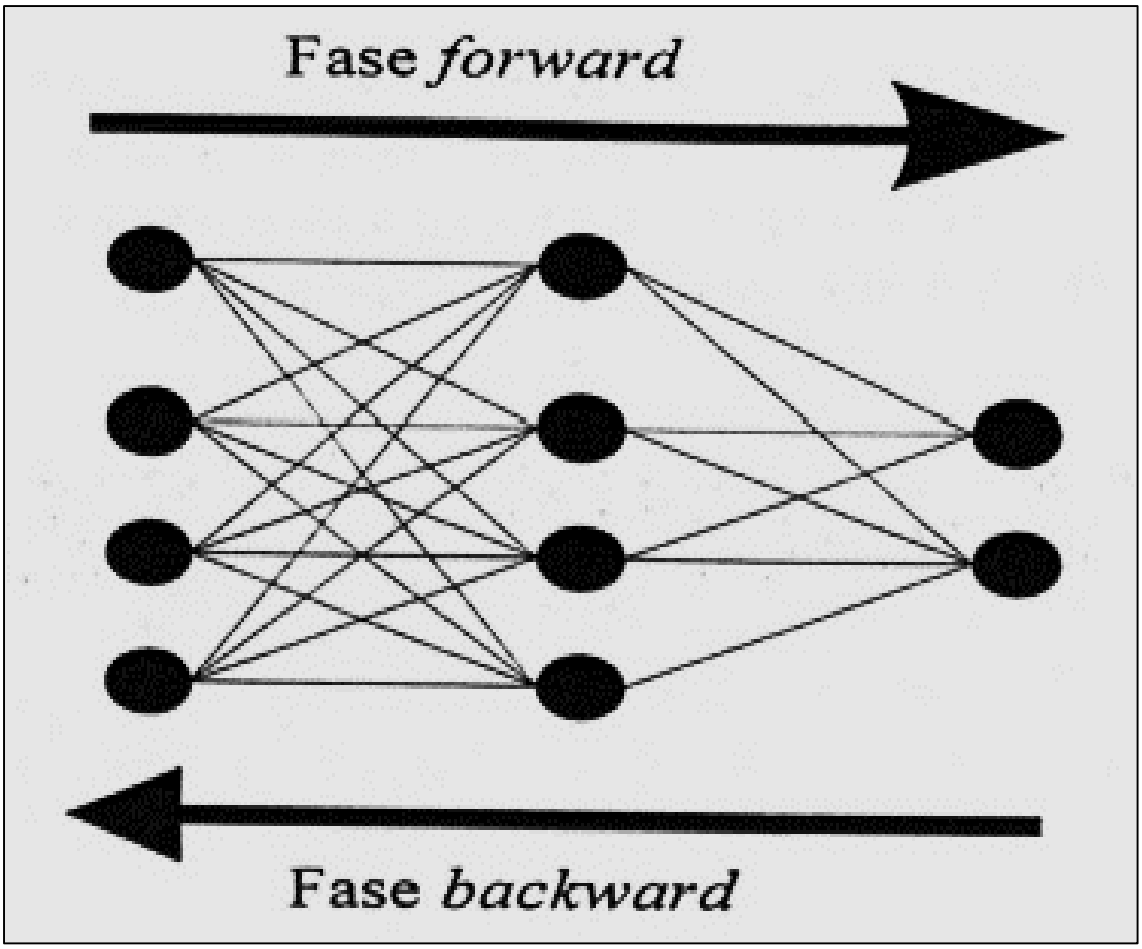

Figura 2.8 - Fluxo do algoritmo *Backpropagation* [3].

O algoritmo *backpropagation* é baseado na regra delta já apresentada, por isso é de chamada de regra delta generalizada. Este algoritmo, propõe uma forma de definir o erro dos nós nas camadas intermediárias, possibilitando o ajuste de seus pesos. A função custo a ser minimizada é uma função de erro ou energia, definida na equação 2.7 como a soma dos erros quadráticos:

$$E = \frac{1}{2} \sum_P \sum_{I=1}^{K} (d_i^P - y_i^P)^2 \tag{2.7}$$

Onde *E* é o erro total, *p* e´ o número de padrões, *k* é o número de unidade de saída, d_i é a *i*-ésima saída desejada e y_i é a *i*-ésima saída gerada pela rede.

Através da equação 2.7 é calculado o erro cometido pela rede [3] e através da regra delta, a variação dos pesos é definida através do gradiente descendente do erro com relação ao peso. Pela derivação da equação 2.7, encontra-se que o ajuste dos pesos é definido pela equação 2.8 ou 2.9:

$$\Delta w_{ij} = \eta \delta_j x_i \tag{2.8}$$

ou:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_j(t) x_i(t) \tag{2.9}$$

Onde:

$\boldsymbol{\eta}$ - taxa da aprendizado da rede; e

$\boldsymbol{\delta}$ - erro calculado da rede.

Através deste algoritmo *back-propagation* procura-se minimizar o erro cometido pela rede, ajustando os pesos e limiares para que correspondam aos pontos de mínimo da superfície de erro. O número de execuções do algoritmo *back-propagation* será determinado de acordo com os critérios de parada escolhidos que poderão ser:

- Encerrar o treinamento após N ciclos;
- Encerrar após o erro quadrático médio ficar abaixo de uma constante α;

- Encerrar quando o percentual de classificações corretas estiver acima de uma constante α;
- Ou combinação dos métodos.

Através deste estudo, conclui-se que as redes *Perceptron* multicamadas são as redes RNAs mais utilizadas em várias aplicações devido sua simplicidade e precisão.

2.11. MODELO DE *HOPFIELD*

Um outro fator chave que aumentou a visibilidade e o respeito para as RNAs, foi o físico Nobel John Hopfield, do Instituto de Tecnologia da Califórnia. Em 1982, juntamente com David Tank, um pesquisador da AT&T, Hopfield desenvolveu um número de redes neurais baseadas nos pesos fixos e nas ativações adaptativas. Estas redes podem servir como redes de memórias associativas e pode ser utilizada para resolver problemas conhecidos como do "Caixeiro Viajante" [5]. Um artigo na revista *Scientific American* ajudou a chamar a atenção popular para redes neurais, com a mensagem de um prêmio Nobel de física que de precisamos estudar cognição humana a fim de fazer máquinas que podem fazer o que os seres humanos fazem.

O modelo de *Hopfield* é um modelo matricial não-linear recorrente, ou seja, as saídas estão ligadas às entradas por um atraso de tempo. Não-linearidades são aplicadas às saídas de cada um dos nós da rede. Essa recorrência implica que a resposta da rede sempre dependa do seu estado no intervalo de tempo anterior.

Através de *Hopfield,* foi possível atribuir um valor de energia para cada estado da rede e esta energia decresce de maneira monotônica à medida que uma trajetória vai em direção a um estado fixo. Estes pontos fixos, são pontos estáveis de energia mínima criados através do processo de treinamento.

Para realizar o armazenamento e recuperação da informação nesta rede, é necessário criar estes pontos fixos e uma regra de atualização que defina a dinâmica da rede. No modelo de *Hopfield,* as saídas dos nós são discretizadas e só podem assumir valores -1 ou +1, pois suas funções de ativação são do tipo degrau. Na figura 2.9, é demonstrado um diagrama esquemático da rede de *Hopfield.*

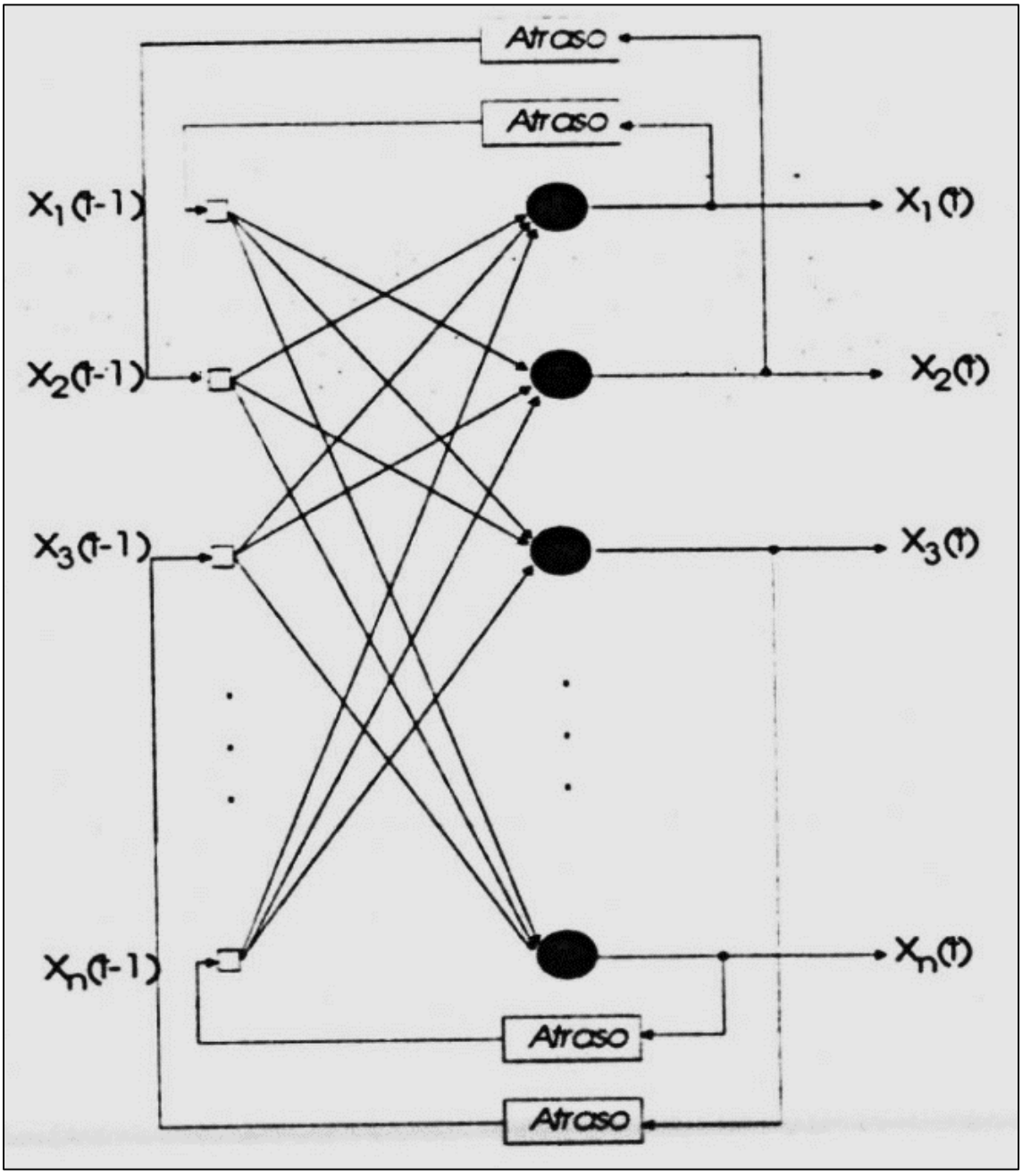

Figura 2.9 - Diagrama esquemático da rede de *Hopfield* [3].

O estudo de uma saída qualquer *i*, no instante de tempo *t+1* é encontrado através da soma ponderada das entradas menos o limiar, conforme apresentado pela equação 2.10. Os elementos da diagonal da matriz dos pesos w_{ij} são nulos, portanto não existe conexão de um nó com ele próprio. Com isto, o próximo estado da saída *i* é função da soma ponderada das saídas dos outros nós no instante de tempo atual.

$$x_i(t+1) = sgn\left(\sum_{j=1}^{p} w_{ij} x_j(t) - \theta_i\right) \quad (2.10)$$

Onde:

θ_i - limiar da unidade *i*; e

x_i - estado da unidade.

Sendo:

$$sgn(x) = \begin{cases} +1 \; se \; x \geq 0 \\ -1 \; se \; x < 0 \end{cases} \quad (2.11)$$

A atualização dos valores dos nós de saída é feita de forma assíncrona, pois as saídas são atualizadas em diferentes tempos. Para implementar esta atualização, cada um dos *k* nós são escolhidas aleatoriamente em diferentes tempos. Assim, este nó escolhido será atualizado de acordo com a regra proposta pela equação 2.11 [3].

Para matriz de pesos qualquer, a condição de estabilidade para um vetor *x* para esta matriz é $sgn\left(\sum_j w_{ij} x_j\right) = x_i$ para todo *i*. Como a constante de proporcionalidade é $\frac{1}{k}$, então $w_{ij} = \frac{1}{k} x_i x_j$, onde *k* é o número de nós da rede.

2.11.1. Exemplo de operação da rede de *Hopfield*

Para exemplificar o funcionamento da rede de Hopfield, consideremos o armazenamento do vetor x=(-1 +1 -1)^T em uma rede. A matriz de pesos obtida é apresentada na Equação 2.12 [3].

$$W = \begin{pmatrix} 0 & -0{,}333 & 0{,}333 \\ -0{,}333 & 0 & -0{,}333 \\ 0{,}333 & -0{,}333 & 0 \end{pmatrix} \quad (2.12)$$

Pode-se verificar que o vetor *x* é estável pois, aplicando a regra de atualização encontra-se que $sgn\left(\sum_j w_{ji} x_j\right) = x_i$ para todo *i*. Da mesma forma, o vetor inverso de x também é estável pois quando se aplica essa regra de atualização para o vetor $\bar{x} = (+1 - 1 + 1)^T$ também manterá os elementos estáveis. Assim, quando a rede foi iniciada em x ou $\bar{x}$, se manterá nestes

pontos indefinitivamente. Estes são os dois pontos fixos criados no espaço de armazenamento do vetor x, que dividirá a rede em duas regiões conhecidas como bacias de atração [3].

Neste exemplo citado, é demostrando o diagrama da máquina de estados dos nós da rede na figura 2.10 para o estado inicial $x(0) = (-1 - 1 - 1)^T$. Observa-se nesta máquina de estados, quando a rede inicia em $x(0)$, a rede sempre se estabiliza nos pontos x ou $\bar{x}$, que são os atratores.

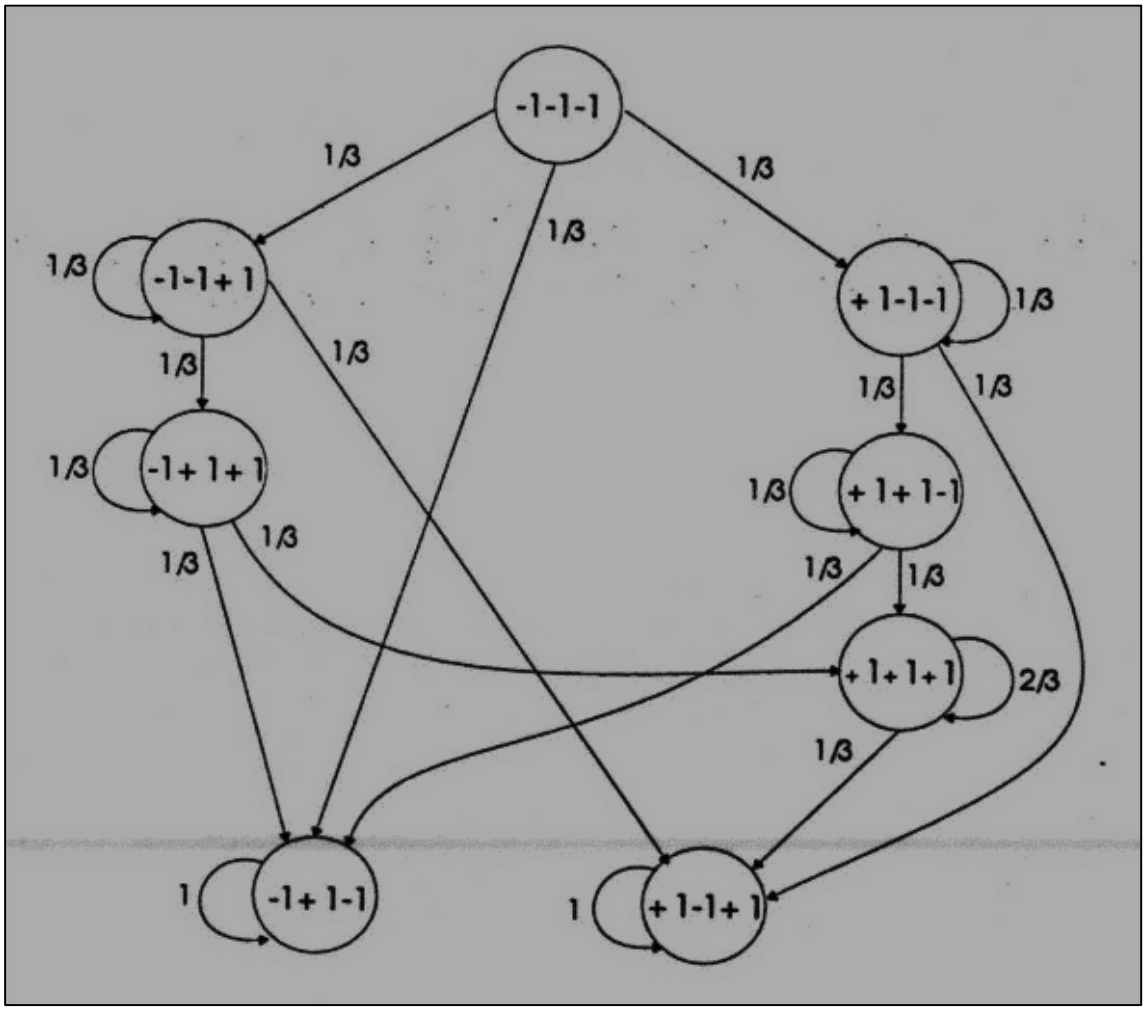

Figura 2.10 - Máquina de estados formada com estado inicial $x(0) = (-1 - 1 - 1)^T$ [3].

O trabalho desenvolvido por *Hopfield* mostra que é possível associar um valor de energia para cada estado da rede. Este valor de energia decresce a medida que os estados da rede evoluem através da regra de atualização das saídas, descrita na equação 2.13.

$$E = -\frac{1}{2}\left(\sum_i \sum_j w_{ij} x_i x_j\right)_{i \neq i} \tag{2.13}$$

Através desta função energia, espera-se pelo armazenamento de x que a função tenha dois mínimos: um em x e outro em $\bar{x}$. Na figura 2.11 pode-se observar os dois mínimos da função energia em x e $\bar{x}$.

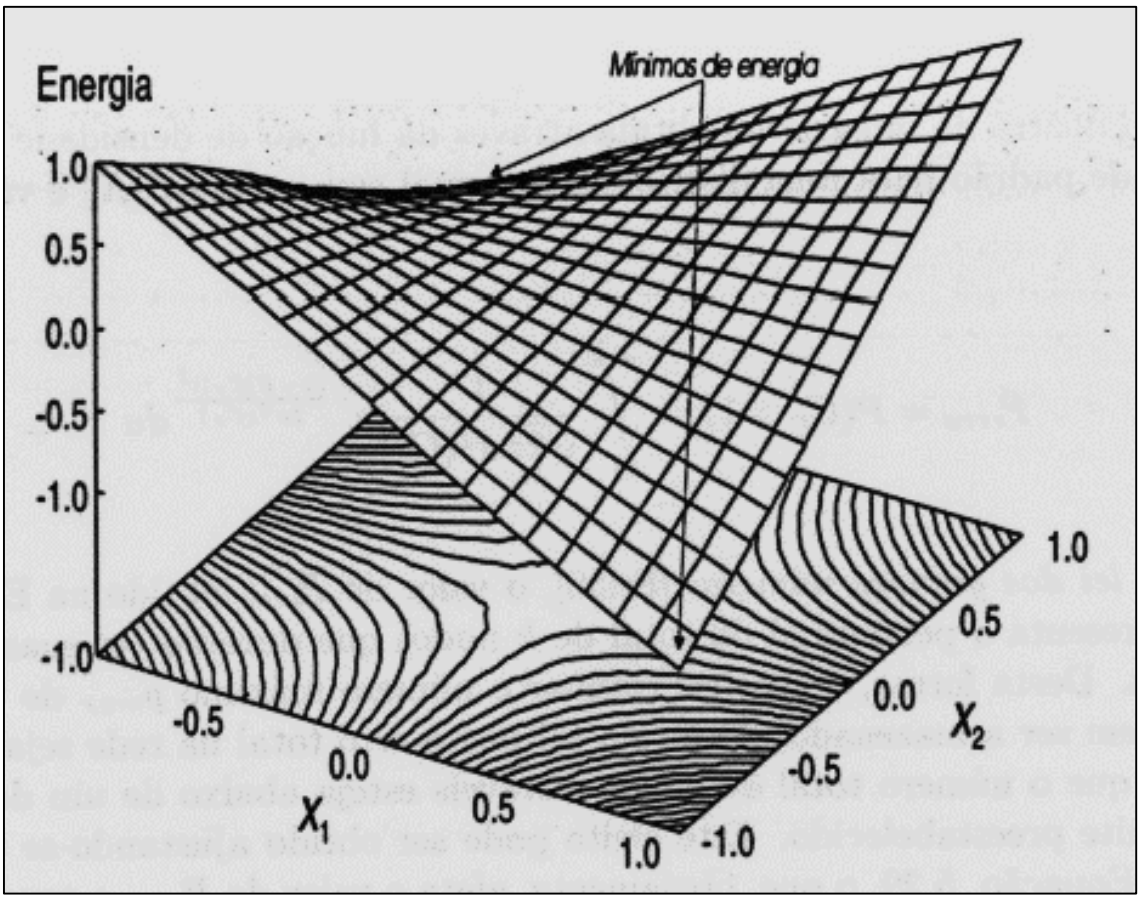

Figura 2.11 - Superfície de energia em uma rede de Hopfield. Os mínimos correspondem a x e $\bar{x}$ [3].

Por fim, com esta rede de *Hopfield*, é possível armazenar vetores com padrões de treinamento ou informações, e através do processo de treinamento encontra-se pontos de estabilidade da rede. Com a rede treinada é possível recuperar as informações que foram armazenadas, isso é possível através da minimização da energia da rede pelo processo de atualização. Por este motivo, a rede de *Hopfield* é conhecida como uma rede de memória associativa que funciona como uma memória endereçada pelo conteúdo.

2.12. CONSIDERAÇÕES FINAIS DESTE CAPÍTULO

Neste capítulo, foi apresentado o conceito de Redes Neurais Artificiais, a evolução dos estudos das RNAs e melhorias das estruturas destas redes. Além disso, foram explicados os principais modelos de Redes Neurais Artificiais conhecidos, iniciando do modelo mais simples de Neurônio Artificial conhecido como Modelo *McCulloch-Pitts*, também foi elencado as estruturas de redes formadas com estes neurônios e os modelos de treinamentos utilizados para aprender padrões. Ao final do capítulo, foi descrito um tipo de rede diferente que utiliza o conceito de memória associativa e energia da rede, que é capaz de armazenar padrões e recuperar posteriormente.

Todas estas redes descritas são utilizadas para solucionar problemas relacionados a reconhecimento e classificações de padrões, e armazenamento e recuperações de padrões. Porém, alguns modelos se mostram mais eficazes e viáveis para solucionar determinados problemas

No próximo capítulo será apresentado um tipo de RNA conhecida como "Máquina de *Boltzmann*" que utiliza um conceito parecido com a rede de *Hopfield* porém faz uso de estados aleatórios e binários das unidades da rede.

CAPÍTULO 3

3. MÁQUINA DE *BOLTZMANN*

3.1. INTRODUÇÃO

No início de 1980 Geoffrey Hinton e Terry Sejnowski desenvolveram o conceito original de uma máquina de Boltzmann. Inspirados no campo da termodinâmica, são sistemas com uma abordagem estocástica que permiti uma fuga de mínimos locais durante o processo de aprendizagem [14] [15].

A Máquina de *Boltzmann* é um tipo de Rede Neural Artificial (RNA) que possui os pesos fixos e é aplicado em problema de otimização. Os estados das unidades usadas são valores binários com requisições probabilísticas de estados [16].

Esta RNA utiliza métodos estocásticos para encontrar parâmetros, onde o acaso desempenha um papel crucial na pesquisa e aprendizagem. A abordagem é permitir a aleatoriedade para ajudar a encontrar bons parâmetros, mesmo em modelos muito complicados. Este método se baseia em conceitos e técnicas da física, especificamente mecânica estatística. Esses métodos estocásticos podem ser preferíveis em problemas complexos [17].

3.2. MÁQUINA DE BOLTZMANN GERAL

A máquina de *Boltzmann* é "uma rede de unidades binárias estocásticos simetricamente acoplados." [18] O sistema tem uma camada de unidades binárias visíveis e uma camada de unidades binárias escondidos. Cada unidade também chamada de neurônio tem uma ligação bidirecional com outras unidades de outras camadas do sistema. O objetivo deste equipamento é o de ser capaz de estimar as probabilidades que ambas as unidades ligadas teriam. Isto é feito, por permitir que a máquina resolva a distribuição próxima do equilíbrio usando tanto a inicialização aleatória e dados orientados das unidades visíveis. A ideia geral é a de ajustar os pesos de modo que as inicializações aleatórias resolvam os estados semelhantes aos que existem no meio ambiente. "A rede será dita ter um modelo perfeito do ambiente se atingir exatamente a mesma distribuição de probabilidade sobre esses dois estados quando está funcionando livremente em equilíbrio térmico com nenhuma entrada do meio ambiente" [14].

Embora este sistema nunca tenha um modelo perfeito, regularidades nos dados podem ser capturados para dar uma estimativa perto da probabilidade real. Esta arquitetura da máquina de Boltzmann é apresentada na Figura 3.1 a seguir.

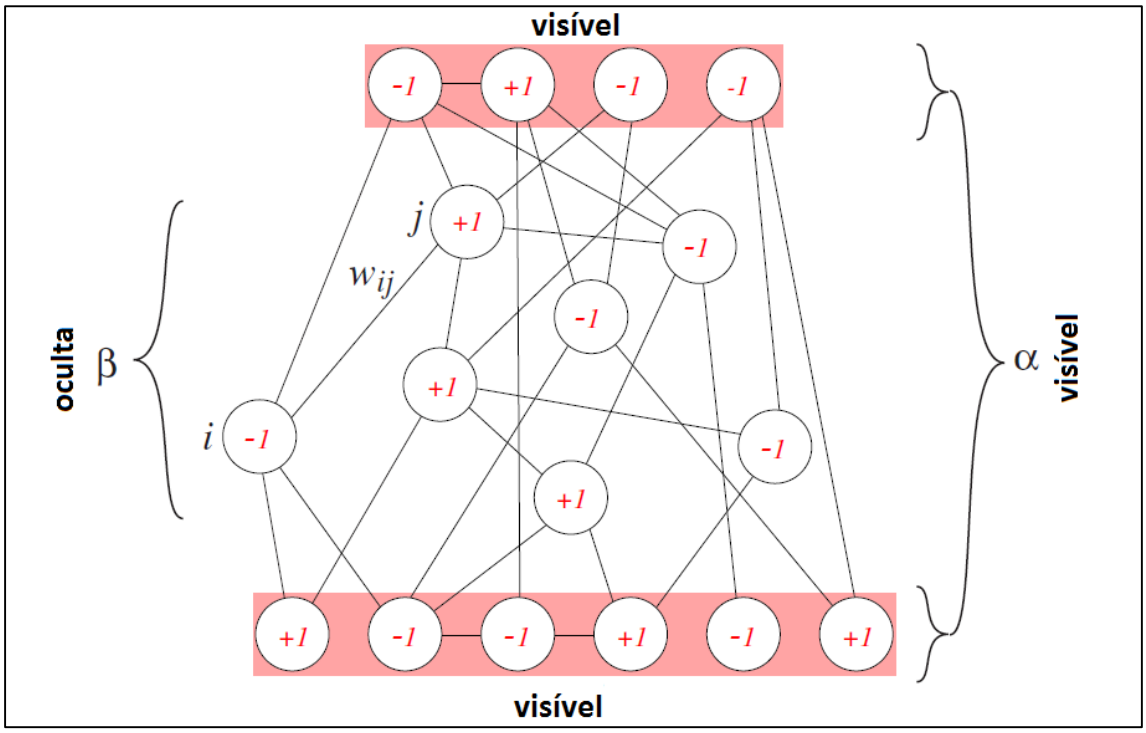

Figura 3.1 - Arquitetura da Máquina de *Boltzmann* [17].

3.3. OTIMIZAÇÃO ESTOCÁSTICA DA REDE

No algoritmo Máquina de *Boltzmann* é necessário resolver um importante problema de otimização. Nesta rede temos um grande número de variáveis ou unidades s_i, $i = 1, ..., N$ onde cada variável pode assumir um dos dois valores discretos ±1. O problema de otimização é encontrar valores das unidades s_i que minimizam a função energia da equação 3.1:

$$E = -\frac{1}{2}\sum_{i,j=1}^{N} w_{ij} s_i s_j \qquad (3.1)$$

Onde:

w_{ij} - pesos entre as conexões dos neurônios ou unidades da rede; e

s_{ij} - unidades ou neurônios da rede.

Os valores w_{ij} que são os pesos de conexões das unidades ou neurônios da rede podem ser tanto negativos quanto positivos. A energia total da rede é encontrada pela soma das interações entre as unidades da rede com os seus respectivos pesos conforme é descrita pela equação 3.1. A configuração mais estável da rede é alcançada através da configuração de menor energia, onde tem-se $\boldsymbol{2^N}$ possíveis configurações de unidades, sendo $\boldsymbol{N}$ igual ao número de unidades ou neurônios da rede [17].

Por isso, em redes com grandes quantidades de unidades, torna-se inviável encontrar uma solução com menor energia através de um algoritmo força-bruta que testa todas configurações possíveis. Isso exigiria bastante tempo e demandaria uma grande carga computacional. Por este motivo, será utilizado a seguir algoritmos que reduzem a energia da rede com um número menor de execuções.

3.4. RECOZIMENTO SIMULADO

No aprendizado de *Boltzmann*, para otimizar ou reduzir a energia do sistema e fugir de soluções com mínimos locais, é utilizado a técnica ou algoritmo chamado de recozimento simulado. No campo da termodinâmica, o recozimento é uma técnica em que o sistema inicia com alta temperatura e a temperatura é abaixada gradualmente até se atingir um equilíbrio e uma baixa energia do sistema [17]. Na termodinâmica, o recozimento é realizado através do aquecimento de uma superfície metálica e um posterior resfriamento lento, de forma que a superfície ficará com uma estrutura mais homogênea [16].

No algoritmo da Máquina de *Boltzmann*, é utilizado este mesmo princípio, porém a temperatura é apenas um parâmetro para se realizar o cálculo da energia da rede e aplicar o algoritmo. Os estados das unidades da rede são alterados várias vezes abaixando a temperatura até se alcançar um limiar escolhido. A cada execução do algoritmo ocorre a atualização dos pesos da rede e são modificados os valores dos neurônios de forma a otimizar a rede.

Será descrito a seguir alguns algoritmos que realizam a técnica do Recozimento Simulado, sendo duas técnicas: uma estocástica e a outra determinística.

3.4.1. Algoritmo de Recozimento Simulado Estocástico

Um dos algoritmos para realizar o recozimento simulado é conhecido com Recozimento Simulado Estocástico. Este algoritmo visa realizar a modificação dos estados ou neurônios da rede de forma a reduzir a energia do sistema, as unidades da rede são representadas pelos

valores de s_i que normalmente assumem valores -1 ou +1 conforme pode-se notar na figura 3.1. Neste algoritmo inicializa-se com uma temperatura *T(1)* alta, com estados aleatórios *Si(1)* e com um valor k_{max} que será a quantidade de vezes que a temperatura *T* será abaixada ou o algoritmo será realizado. Seleciona-se aleatoriamente um nó *i* da rede que poderá ser atualizado de acordo com a regra a seguir. Suponha-se que um estado seja $s_i = +1$, então calcula-se a energia E_a do sistema nesta configuração; depois recalcula-se a energia E_b para o novo estado $s_i = -1$. Se este estado candidato tiver energia menor, aceita-se esta mudança de estado. Caso a energia seja maior, aceita-se a mudança com a probabilidade igual a expressão 3.2 [17]:

$$e^{-\Delta E_{ab}/T} \tag{3.2}$$

Onde:

$\Delta E_{ab} = E_b - E_a$; e

T - Temperatura.

Este algoritmo realiza a seleção e teste várias vezes dos nós *i* da rede sempre abaixando a temperatura. O recozimento simulado termina quando a temperatura está muito baixa (definido através do valor de *k*). A figura 3.2 descreve os passos do algoritmo explicado.

```
1  Inicializar T(k), k_max, s_i(1), w_ij para i, j = 1, ..., N
2       k ← 0
3       do k ← k + 1
4          do selecionar nó i aleatoriamente; estado s_i
5                E_a ← − (1/2) Σ_{i,j=1}^{N} w_ij s_i s_j
6                E_b ← E_a
7                if E_b < E_a
8                  then s_i ← -s_i
9                     else if e^{-(E_b − E_a)/T(k)} > rand (0,1)
10                          then s_i ← -s_i
11                        until
12             until k = k_max ou encontrar critério de parada
13           return E, s_i, for i=1, …, N
14         end
```

Figura 3.2 - Algoritmo de recozimento simulado estocástico [17].

Caso o resfriamento seja suficiente, o sistema terá grande probabilidade de estar em baixo estado de energia, atingindo o mínimo global de energia [17].

3.4.2. Recozimento Simulado Determinístico

Um outro algoritmo utilizado para realizar o recozimento é chamado de determinístico porque em princípio é possível deterministicamente resolver as equações de forma a encontrar novos valores para os estados s_i da rede, quando a temperatura é reduzida.

Neste método de recozimento determinístico, seleciona-se aleatoriamente um nó *i* da rede, depois calcula-se as forças exercidas dos nós conectados a este nó *i* e este somatório é chamado l_i conforme a equação 3.3 a seguir:

$$l_i = \sum_j w_{ij} s_j \tag{3.3}$$

Onde:

w_{ij} - Pesos que conectam os nós i e j; e

S_i - Valor do nó *i* seleccionado.

O valor de s_i a ser atualizado dependerá deste somatório l_i e da temperatura em andamento da rede. Sendo aplicada a equação 3.4, calcula-se a tangente hiperbólica de l_i/T.

$$s_i = f(l_i, T) = \tanh[l_i/T] \tag{3.4}$$

Onde:

w_{ij} - Pesos que conectam os nós *i* e *j*;

S_i - Valor do nó *i* seleccionado; e

l_i - Somatório explicado na equação 3.

Os novos valores de S_i selecionados serão substituidos na rede segundo os valores determinados pela equação 3.4. A figura 3.3 apresenta as etapas descritas deste algoritmo.

```
1  Inicializar T(k), k_max, w_ij, s_i(1), i, j = 1, ..., N
2       k ← 0
3       do k ← k + 1
4            selecionar nó i aleatoriamente; estado s_i
5               l_i = Σ_j w_ij s_j
6               s_i = f(l_i, T) = tanh[l_i/T]
7            until k = k_max ou encontrar critério de
convergência
8     return E, s_i, i = 1, ..., N
9  end
```

Figura 3.3 - Algoritmo de recozimento simulado determinístico [17].

Ao final deste processo, também espera-se encontrar valores de s_i que reduzam a energia total do sistema.

3.5. APRENDIZAGEM DE BOLTZMANN

No algoritmo de *Boltzmann*, o recozimento é utilizado para diminuir a energia do sistema, podendo ser qualquer um dos dois métodos explicados. Porém para realizar reconhecimento de padrões é necessário manter fixo alguns valores dos neurônios S_i e realizar o recozimento dos demais, este algoritmo será descrito a seguir.

3.5.1. Algoritmo de Aprendizagem de *Boltzmann*

No algoritmo de aprendizagem de *Boltzmann*, é necessário inicializar com os valores de taxa de aprendizagem η, temperatura inicial *T(k) e* pesos w_{ij}. Inicialmente define-se aletoriamente os estados S_i. O próximo passo será manter fixo os neurônios de entrada e saída a serem treinados, e realizar o recozimento dos demais neurônios que são os neurônios da camada oculta. No final deste passo, é necessário abaixar a temperatura *T* e calcular o produto $s_i s_j$.

Novamente, define-se de forma aleatória os estados s_i. Inicia-se então outro processo de recozimento, desta vez mantendo fixo apenas a camada de entrada da rede e realizando o recozimento da camada oculta e camada de saída. Abaixa-se novamente a temperatura *T* do sistema e calcula-se o produto $s_i s_j$.

Através dos produtos $s_i s_j$ calculados nos dois processos, da taxa de aprendizagem η e da temperatura T, atualiza-se os pesos da rede seguindo a seguinte equação 3.5.

$$w_{ij} = w_{ij} + \eta/T \left[\left[s_i s_j \right]_{\alpha^i \alpha^o \, fixas} - \left[s_i s_j \right]_{\alpha^i \, fixa} \right] \tag{3.5}$$

Onde:

w_{ij} - Pesos que conectam os nós i e j;

S_i - Valor do nó *i* selecionado;

η - Taxa de aprendizagem;

T - Temperatura;

α^i - Unidades de entrada da rede; e

α^o - Unidades de saída da rede.

Este processo de atualização será realizado *k* vezes, sendo *k* a quantidade de vezes que a temperatura é reduzida. A figura 3.4 descreve este algoritmo explicado. Ao final do processo, a rede terá os pesos w_{ij} atualizados e os valores s_i dos neurônios terá valores que tendem abaixar a energia do sistema e manter a rede estável.

```
1 Inicializar Ɗ, η, T(k), w_ij, i, j = 1, ... , N
2       do Selecionar aleatoriamente padrão de treinamento x
3        Estados aleatórios s_i
4        Recozimento da rede com entradas e saídas fixas
5        No final, abaixar T, calcular [s_i s_j]_{α^i α^o fixos}
6        Estados aleatórios s_i
7           Recozimento com entradas fixas e saídas livres
8           No final, abaixar T, calcular [s_i s_j]_{α^i fixos}
9           w_ij ← w_ij + η/T [[s_i s_j]_{α^i α^o fixos} − [s_i s_j]_{α^i fixos}]
10      until k = k_max ou encontrar critério de convergência
11   return w_ij
12  end
```

Figura 3.4 - Algoritmo determinístico de *Boltzmann* [17].

Neste processo, a rede formada terá valores com grande probabilidade de realizar reconhecimento de padrões. Assim, com os valores de entrada e saída treinados na rede, será possível realizar a classificação de padrões de acordo com a entrada.

3.6. MÁQUINA DE *BOLTZMANN* RESTRITA

Nos últimos anos, um dos modelos gráficos probabilísticos de redes neurais estocásticas que tem ganhado popularidade, são as Máquinas de *Boltzmann* Restritas (RBM). Essas redes têm aplicação em classificação, processamento e geração de imagens, aprendizagem de padrões, modelagem acústica, entre outros. RBMs são casos especiais de redes do tipo Máquinas de *Boltzmann* (BM) [19]. A diferença entre a BM e RBM está no gradiente necessário para a aprendizagem utilizando máxima verossimilhança. Mesmo com a ajuda do recozimento simulado da Máquina de *Boltzmann*, este processo de aprendizagem é considerado lento para certas aplicações. Por isso o aprendizado pode ser mais eficiente na Máquina *Boltzmann* Restrita, pois este algoritmo não possui conexões entre unidades de mesma camada [18].

A Máquina de *Boltzmann* Restrita é uma simples Máquina de *Boltzman* com topologia restrita, este algoritmo possui uma camada de unidades visíveis e uma camada de unidades ocultas, porém não há conexões entres as unidades de mesma camada como: visíveis-visíveis e ocultas-ocultas [20]. A figura 3.5 ilustra a estrutura gráfica desta rede.

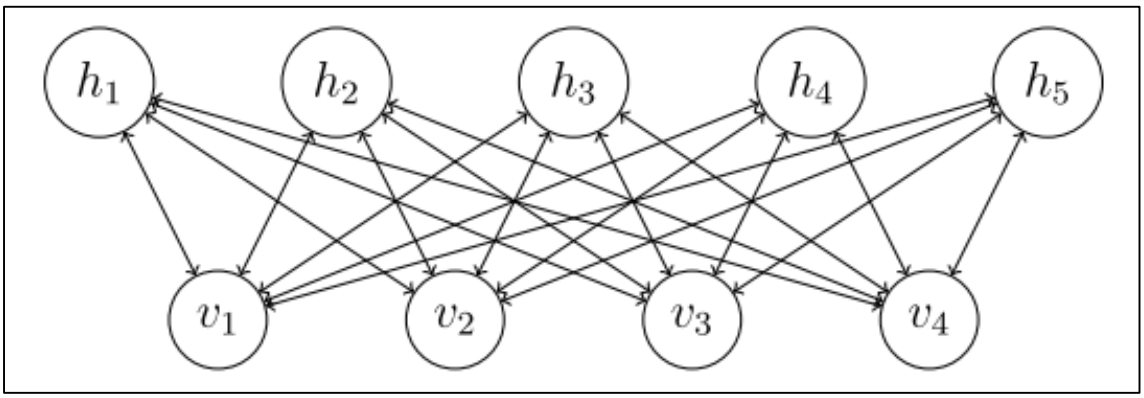

Figura 3.5 - Estrutura da Máquina de *Boltzmann* Restrita com 4 unidades visíveis e 5 ocultas.

3.7. ESTIMAÇÃO DOS PARÂMETROS DE UMA RBM

Em uma RBM, também é necessário ajustar os parâmetros da rede como os pesos das conexões, através de uma distribuição de probabilidades. Porém na RBM são utilizados métodos diferentes de otimização das que são utilizados na Máquina de *Boltzmann*. Na RBM, os estados ocultos, são condicionalmente independentes e os estados visíveis e a distribuição dos estados s_j é dada pela função logística 3.6:

$$p(s_j = 1) = \frac{1}{1+e^{(-\sum_i w_{ij} s_i)}} \tag{3.6}$$

A função energia *E(v,h)* de uma RBM é representada pela função 3.7 [21]:

$$E(v,h) = -\sum_i b_i v_i - \sum_j c_j h_j - \sum_{i,j} w_{ij} v_i h_j \tag{3.7}$$

Onde w_{ij} representam os pesos entre a camada de unidades visíveis *i* e camada de unidades ocultas *j*. Os parâmetros *c* e *b* podem representar os *bias* das camadas ocultas e visíveis respectivamente. Os pesos w_{ij} podem ser estimados utilizando máxima verossimilhança. Esta forma de estimar os parâmetros pode ajudar minimizar a energia de distribuição dos estados e aumentar a energia com estados menos prováveis. A probabilidade de distribuição conjunta desta rede é representada pela equação 3.8:

$$P(v,h) = \frac{\exp(-E(v,j))}{Z} \tag{3.8}$$

Onde:

$z = \sum_{v,h} \exp(-E(v,h))$ - função partição de *Boltzmann*.

Pode ser calculada a distribuição marginal de unidades visíveis pelo somatório de todas possíveis unidades ocultas conforme a equação 3.9 a seguir.

$$P(v) = \frac{1}{Z} \sum_h \exp(-E(v,h)) \tag{3.9}$$

Além disso, quando as unidades da rede possuem valores binários, a probabilidade de ativação dos neurônios é representada pela equação 3.10 a seguir [21].

$$\begin{cases} P(v_i = 1|h) = sigm(b_j + W_j'h) \\ P(h_i = 1|v) = sigm(c_j + W_i v) \end{cases} \tag{3.10}$$

3.8. ALGORITMO DE APRENDIZAGEM DA MÁQUINA DE BOLTZMANN RESTRITA

Através das equações da energia da rede e da probabilidade dos estados das unidades, pode-se utilizar a função verossimilhança para estimar os pesos da rede. Nas equações a seguir é aplicado a derivada da função verossimilhança em relação aos pesos da rede de forma a ajustar os valores dos pesos. O treinamento da RBM é geralmente baseado na maximização do gradiente da função verossimilhança baseado nos parâmetros da RBM dado os valores de treinamento [19].

$$E(v, h|\theta) = \sum_{ij} w_{ij} v_i h_j + \sum_i b_i v_i + \sum_j b_j h_j \tag{3.11}$$

$$p(v|\theta) = \prod_{n=1}^{N} \sum_h p(v, h|\theta) = \prod_{n=1}^{N} \frac{\sum_h \exp(-E(v,h|\theta))}{\sum_{v,h} \exp(-E(v,h|\theta))} \tag{3.12}$$

$$\log \rho(v|\theta) = \sum_{n=1}^{N} (\log \sum_h \exp\big(-E(v, h|\theta)\big) - \log \sum_{v,h} \exp\big(-E(v, h|\theta)\big)) \tag{3.13}$$

$$\frac{\partial \log \rho(v|\theta)}{\partial w_{ij}} = \sum_{n=1}^{N} \big[v_i \sum_h h_j \rho(h|v) - \sum_{v,h} v_i h_j \rho(v, h)\big] \tag{3.14}$$

$$= E_{data}\big[v_i h_j\big] - E_{model}\big[v'_i h'_j\big] \equiv < v_i h_j >_{data} - < v'_i h'_j >_{model} \tag{3.15}$$

Pela sequência das equações 3.11 a 3.14, encontra-se a variação dos pesos da rede na equação 3.15. Como $< v'_i h'_j >_{model}$ representa o i-ésimo valor dos estados, ele exigi uma grande carga computacional para ser estimado, por isso, aplica-se uma Cadeia de *Markov* de Monte Carlo, a amostragem de *Gibbs,* para estimá-lo. Amostragem de *Gibbs* é um algoritmo simples de Cadeia de *Markov* para produzir amostragens de distribuição de probabilidade conjunta de múltiplas variáveis aleatórias. A ideia é construir uma cadeia de *Markov* para atualizar cada variável baseada na distribuição condicional dado os demais estados [19].

Na figura 3.6 é apresentado a estrutura de aprendizagem e atualização de valores da Máquina de *Boltzmann* Restrita.

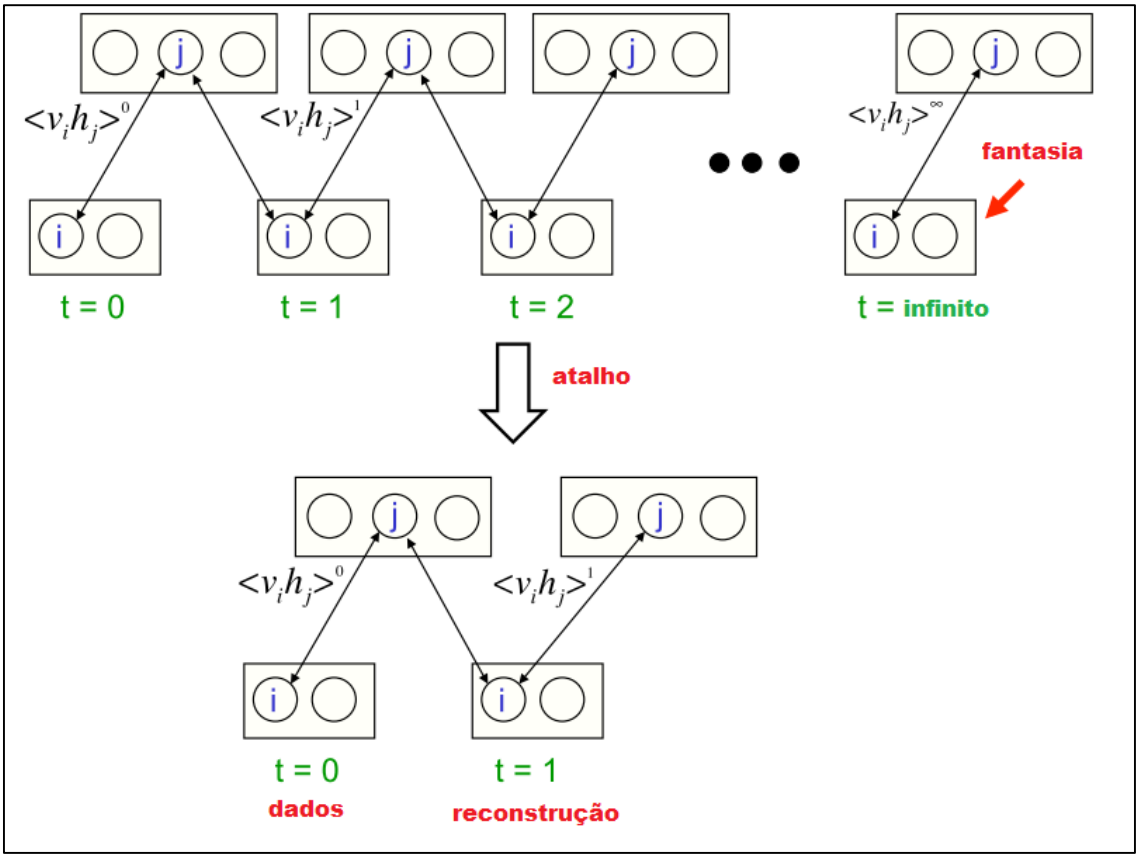

Figura 3.6 - Estrutura da Máquina de *Bolztmann* Restrita com 4 unidades visíveis e 5 ocultas.

Os passos executados neste algoritmo da figura 3.6 para executar a amostragem de *Gibbs* são enumerados de 1 a 5 a seguir:

1. Inicia-se com um vetor de treinamento nas unidades visíveis.
2. Atualiza todas as unidades ocultas em paralelo conforme a equação 3.10.
3. Atualiza todas as unidades visíveis em paralelo utilizando também a equação 3.10 para obter a "reconstrução".
4. Atualiza novamente as unidades ocultas.
5. Os pesos treinados serão encontrados aplicando a equação 3.16 a seguir.

$$\Delta w_{ij} = \in \left(< v_i^0 h_j^0 > - < v_i^1 h_j^1 > \right) \tag{3.16}$$

Além disso, neste algoritmo de aprendizagem da RBM, como já mencionado, busca-se aproximar os valores da função máxima verossimilhança 3.16. As fases de treinamento da rede são chamadas de fase positiva (*wake*) quando se utiliza os valores de entrada da rede, e fase negativa (*sleep*) que se utiliza as unidades ocultas encontradas na fase anterior.

$$\frac{\partial \log \rho(v)}{\partial w_{ij}} \approx \frac{1}{N} \sum_{n=1}^{N} \left[v_i^{(n)} h_j^{(n)} - v'^{(n)}_i h'^{(n)}_j \right] \tag{3.17}$$

Onde:

$v_i^{(n)}$ – valor da i-ésima unidade visível para o n-ésimo treinamento;

$h_j^{(n)}$ – é valor da j-ésima unidade oculta;

$v'^{(n)}_i$ – valor amostrado da i-ésima unidade visível ou dados negativos gerados baseado em *h(n)* e *w*; e

$h'^{(n)}_j$ – valor amostrado da j-ésima unidade oculta ou atividades ocultas negativas geradas com base em *v(n)* e w_{ij}.

Através deste algoritmo que é chamado *wake-sleep* é possível atualizar todos os nós da rede treinada. Na figura 3.7 a seguir é mostrado de forma determinística com ocorre a atualização dos valores das unidades da rede.

Algoritmo Wake-sleep

I. Fase positiva ("*wake*") (fixar unidades visíveis com dados):

- Usar dados de entrada para gerar atividades ocultas:

$$h_j = \frac{1}{1 + \exp(-\sum_i v_i w_{ij} - b_j)}$$

- Amostrar estados ocultos para distribuição de Bernoulli:

$$h_j \leftarrow \begin{cases} 1, & se\ h_j > rand(0,1) \\ 0, & outros \end{cases}$$

II. Fase negativa ("*sleep*") (não fixar as unidades visíveis dos dados):

- Usar h_j para gerar dados negativos:

$$v'_i = \frac{1}{1 + \exp(-\sum_j w_{ij} h_j - b_i)}$$

- Usar dados negativos v'_i para gerar atividades ocultas negativas:

$$h'_j = \frac{1}{1 + \exp(-\sum_j v'_i w_{ij} - b_j)}$$

Figura 3.7 - Algoritmo de treinamento *wake-sleep* da Máquina de *Boltzmann* Restrita.

Ao final deste algoritmo, todos os valores das unidades visíveis e ocultas da rede serão atualizados.

3.9. APRENDIZAGEM DIVERGÊNCIA COMPARATIVA CD-K

Voltando a atenção na parte inferior da figura 3.6, nota-se que é utilizado um atalho no algoritmo do treinamento da RBM, para encontrar o gradiente da função verossimilhança em relação aos pesos. Este algoritmo utilizado é chamado de aprendizagem Divergência Comparativa CD-k, ao invés de realizar várias atualizações das unidades da rede, executa-se apenas uma atualização para encontrar a variação dos pesos conforme a equação 3.18 [22].

$$\frac{\partial \log \rho(v^0)}{\partial w_{ij}} = < h_j^0 (v_i^0 - v_i^1) > + < v_i^1 \left(h_j^0 - h_j^1\right) > + < h_j^1 (v_i^1 - v_i^2) > + \cdots$$
$$= < v_i^0 h_j^0 > - < v_i^\infty h_j^\infty > \approx < v_i^0 h_j^0 > - < v_i^1 h_j^1 > \qquad (3.18)$$

Para isto iniciamos a cadeia de *Markov* com um exemplo do treinamento, de modo que a cadeia já estará perto de ter convergido para distribuição final. As amostras são colhidas depois de apenas k-passos de amostragem de *Gibbs*. Na prática, k = 1 tem sido demonstrado

que funciona surpreendentemente bem. Apesar de não esperar a convergência, esta técnica apresenta resultados muito bons e um custo computacional muito menor, justificando seu uso [22].

Ao final deste processo de amostragem e treinamento, é possível reconstruir os dados treinados da rede e utilizar este algoritmo para solucionar problemas relacionados a reconhecimento de padrões.

3.10. CONSIDERAÇÕES FINAIS DESTE CAPÍTULO

Neste capítulo, foi apresentado dois tipos de Redes Neurais Artificiais que utilizam um modelo probabilístico e estocástico para modelagem da rede. Foi introduzido a ideia do algoritmo Máquina de *Boltzmann* que utiliza o conceito das redes de *Hopfield* no qual as redes possuem uma energia associada.

No primeiro modelo de rede estocástica apresentada chamada Máquina de *Boltzmann* Geral, o ajuste da rede é feito de forma minimizar a energia total da rede através do ajuste dos pesos e dos estados dos neurônios. Para realizar este ajuste dos pesos da rede, é utilizado o algoritmo conhecido como recozimento simulado que ajuda a minimizar a energia da rede com um número menor de execuções.

Posteriormente foi apresentada a rede Máquina de *Boltzmann* Restrita, que possui restrições nas conexões da rede, não existindo conexões entre neurônios de mesma camada. Nesta rede, a modelagem da rede é encontrada através de um algoritmo que utiliza uma Cadeia de *Markov* de Monte Carlo chamada Amostragem de *Gibbs* para estimar os parâmetros os pesos rede. Este algoritmo encontra a máxima verossimilhança da derivada da função probabilidade dos estados da rede em relação aos pesos das conexões. Para agilizar este processo de amostragem é utilizada a aprendizagem Divergência Comparativa CD-k, que modela os pesos da rede de forma mais rápida.

Ambos modelos de Redes Estocásticas, são utilizados para encontrar estados da rede com grande probabilidade de selecionar parâmetros treinados e reconstruir dados. Estas redes podem ser utilizadas para problemas de reconhecimento de padrões pois são capazes de reproduzir decisões baseadas em seu treinamento. Porém, o algoritmo Máquina de *Boltzmann* Restrita é um algoritmo que exige menos execuções convergindo mais rápido.

CAPÍTULO 4

4. SIMULAÇÕES COMPUTACIONAIS DOS ALGORITMOS: REDE NEURAL ARTIFICIAL *BACKPROPAGATION*, MÁQUINA DE *BOLTZMANN* GERAL E MÁQUINA DE *BOLTZMANN* RESTRITA PARA RECONHECIMENTO DE PADRÕES BINÁRIOS

4.1. INTRODUÇÃO

Neste capítulo será apresentada as simulações computacionais dos algoritmos utilizados para reconhecimento de padrões. Após apresentados, nos capítulos anteriores, alguns algoritmos que são capazes de treinar, classificar e reconstruir padrões; foram testados nesta pesquisa os principais algoritmos descritos. O algoritmo RNA *Backpropagation* apresentado na seção 2.10 deste trabalho foi utilizado para classificar padrões e comparar a performance dos dois algoritmos que utilizam o conceito Máquina de *Bolzmann* apresentados no capítulo 3 deste trabalho.

Nas simulações computacionais realizadas, foram utilizados os mesmos valores de entradas da rede para os três algoritmos de reconhecimento de padrões. Foi utilizado o software *Matlab* para criação dos algoritmos: RNA *Backpropagation*, Máquina de *Boltzmann* Geral e Máquina de *Boltzmann* Restrita.

4.2. CLASSIFICAÇÃO DE PADRÕES DE 7 SEGMENTOS

Inicialmente foi utilizado como entrada valores binários com combinações formando 10 padrões de um display de 7 segmentos, conforme pode ser visto na figura 4.1 a seguir.

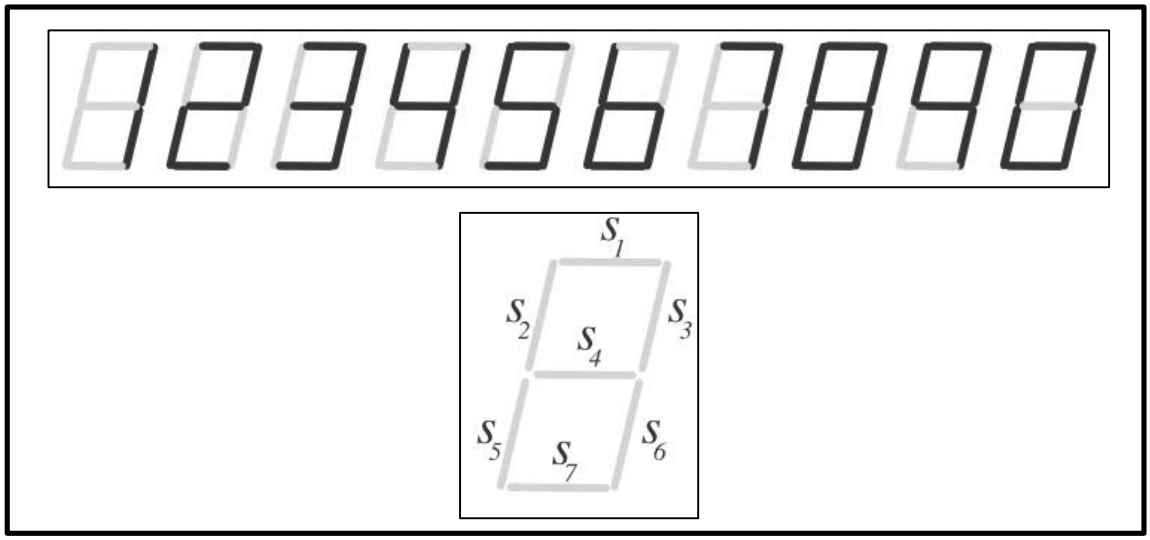

Figura 4.1 - Padrões formados pelo display de 7 segmentos [17].

Conforme é apresentado na figura 4.1, os 10 dígitos existentes são formados de acordo com a combinação dos segmentos destacados na cor preta. Para exemplificar, o primeiro dígito da figura, dígito "1", apresenta os segmentos S_3 e S_6 destacados na cor preta e os demais segmentos na cor cinza. Nas redes neurais artificiais criadas neste trabalho, que serão tratadas nos próximos tópicos, os segmentos pretos são representados pelo valor +1 e os segmentos cinza por -1. Desta forma, as entradas das redes apresentam 10 padrões distintos com 7 valores binários cada, conforme é representado na figura 4.2 a seguir.

```
x =

     1     1     1    -1     1     1     1
    -1     1    -1    -1     1    -1    -1
     1    -1     1     1     1    -1     1
     1    -1     1     1    -1     1     1
    -1     1     1     1     1    -1    -1
     1     1    -1     1    -1     1     1
    -1    -1    -1     1     1     1     1
     1    -1     1    -1    -1     1    -1
     1     1     1     1     1     1     1
     1     1     1     1    -1     1    -1
```

Figura 4.2 - Padrões formados pelo display de 7 segmentos.

Com estes padrões de entrada a serem utilizados para testes de classificação, as redes possuirão 7 neurônios na camada de entrada e 7 neurônios na camada de saída. Isto porque os algoritmos treinados deverão reconstruir na saída os mesmos valores da utilizados na entrada.

4.3. CLASSIFICAÇÃO DE PADRÕES DE SEQUÊNCIAS BINÁRIAS

Após realizados os testes com os dados dos padrões de 7 segmentos do item anterior, foram testadas entradas com quantidades maiores de dados. Os novos padrões utilizados para testes nas simulações, foram sequências numéricas formadas por valores binários “-1” e “+1” e determinados pela quantidade bits escolhidos conforme a figura 4.3 a seguir.

```
>> gera_seq1(5)

ans =

    -1    -1    -1    -1    -1
     1    -1    -1    -1    -1
    -1     1    -1    -1    -1
     1     1    -1    -1    -1
    -1    -1     1    -1    -1
     1    -1     1    -1    -1
    -1     1     1    -1    -1
     1     1     1    -1    -1
    -1    -1    -1     1    -1
     1    -1    -1     1    -1
    -1     1    -1     1    -1
     1     1    -1     1    -1
    -1    -1     1     1    -1
     1    -1     1     1    -1
    -1     1     1     1    -1
     1     1     1     1    -1
    -1    -1    -1    -1     1
     1    -1    -1    -1     1
    -1     1    -1    -1     1
     1     1    -1    -1     1
    -1    -1     1    -1     1
     1    -1     1    -1     1
    -1     1     1    -1     1
     1     1     1    -1     1
    -1    -1    -1     1     1
     1    -1    -1     1     1
    -1     1    -1     1     1
     1     1    -1     1     1
    -1    -1     1     1     1
     1    -1     1     1     1
    -1     1     1     1     1
     1     1     1     1     1
```

Figura 4.3 - Padrões formados por sequências de 5 bits binários.

Neste exemplo simulado na figura 4.3, a quantidade de *bits* escolhida para gerar as sequências foi 5, gerando assim um total de $2^5 = 32$ padrões diferentes de entradas; também foi utilizado neste trabalho simulações com 6 *bits* correspondendo a $2^6 = 64$, 7 *bits* correspondendo a $2^7 = 128$ e 8 *bits* correspondendo a $2^8 = 256$ padrões de entradas e saídas. Estes padrões correspondem a todas as possíveis combinações com o total de *bits* escolhidos. Desta forma, foi possível realizar simulações com uma quantidade maior de padrões, testando assim a capacidade das redes em classificar padrões.

4.4. SIMULAÇÕES COM RNA *BACKPROPAGATION*

Neste trabalho, inicialmente criou-se uma RNA *Backpropagation* para testar a capacidade de classificação de padrões. Na figura 4.4 a seguir, é representada a estrutura da rede criada. Esta rede possui 6 neurônios na camada de entrada, 8 neurônios na camada oculta, e 6 neurônios na camada de saída.

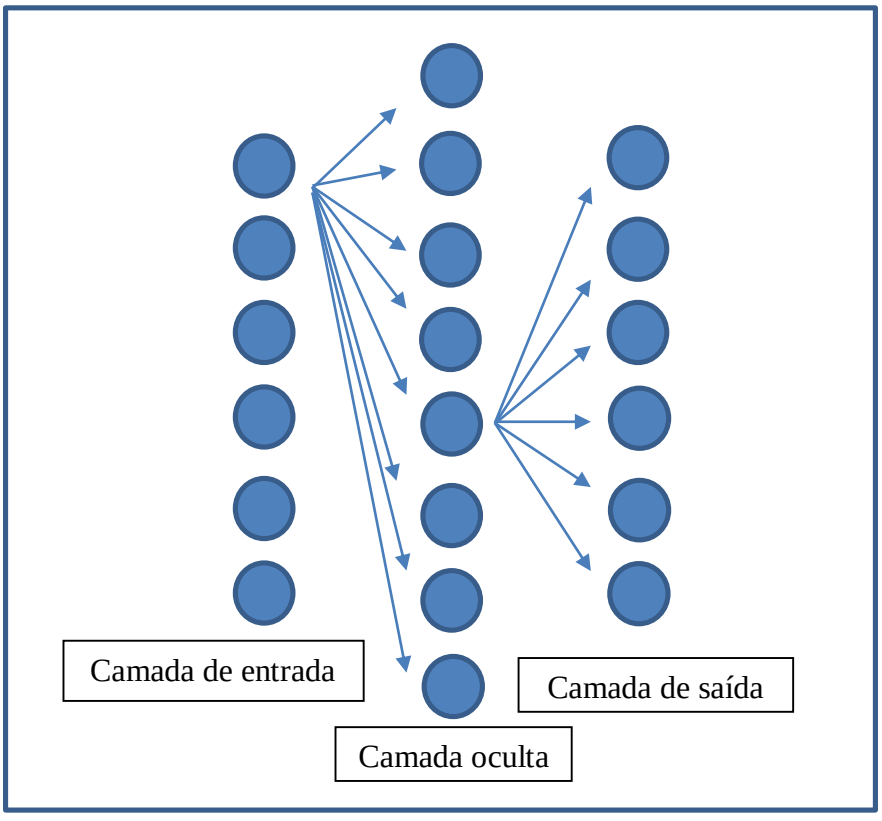

Figura 4.4 - Estrutura de RNA *Backpropagation*.

Para realizar um dos testes foi utilizado $2^6 = 64$ padrões diferentes de entrada sendo cada padrão com 6 valores binários. O treinamento da rede foi de forma a igualar valores dos padrões

da camada de entrada à camada de saída. Para testar a acurácia da rede testou-se o percentual de acerto da rede treinada.

Além disso, foi inserido um ruído aleatório de 0,3 ou 30 % da potência do sinal conforme é descrito através da função da equação 4.1 que gera ruídos aleatórios na rede de forma a testar a resposta da rede nestas condições. Desta forma, calculou-se também o percentual de acertos nestas condições.

$$x(i,:) + 0.3 * (\text{rand} - .5); \qquad (4.1)$$

Primeiramente, a rede foi testada considerando como entrada os padrões de 7 segmentos e, depois, considerando como entrada a sequência binária de 6 *bits*. Na sequência de gráficos das figuras 4.5 a 4.12 a seguir, são plotadas simulações com diferentes condições de entrada, realizadas variando a quantidade de épocas de 0 a 200. Nestes gráficos é plotado o gráfico do percentual de acerto por número de épocas e, em seguida, é plotado o tempo de execução do código por número de épocas. Também são plotados gráficos com inserção de ruído já mencionado. Na legenda destes gráficos, é descrito o tipo de entrada utilizada nas simulações realizadas.

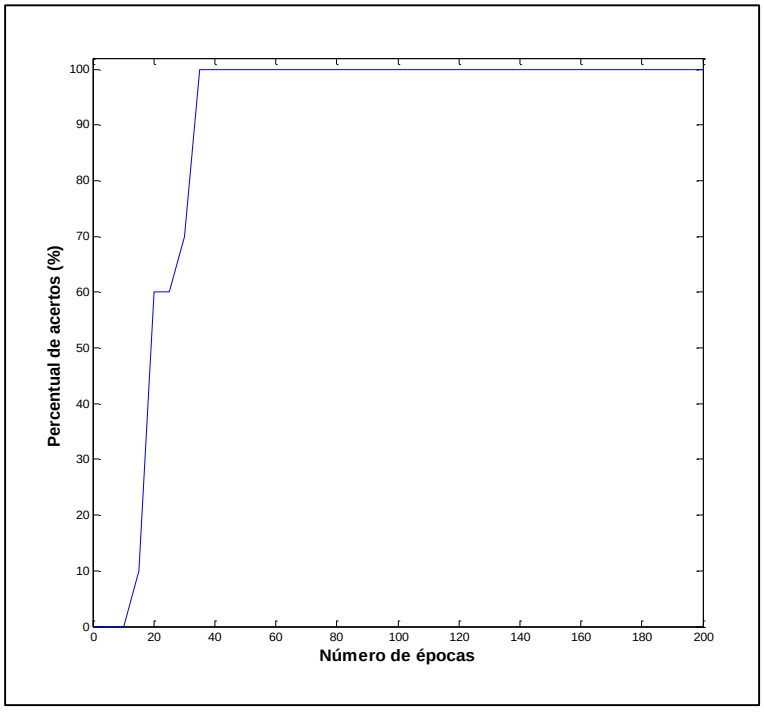

Figura 4.5 - Gráfico do percentual de acertos por número de épocas para a rede RNA *Backpropagation* sem ruído utilizando na entrada dígitos de 7 segmentos.

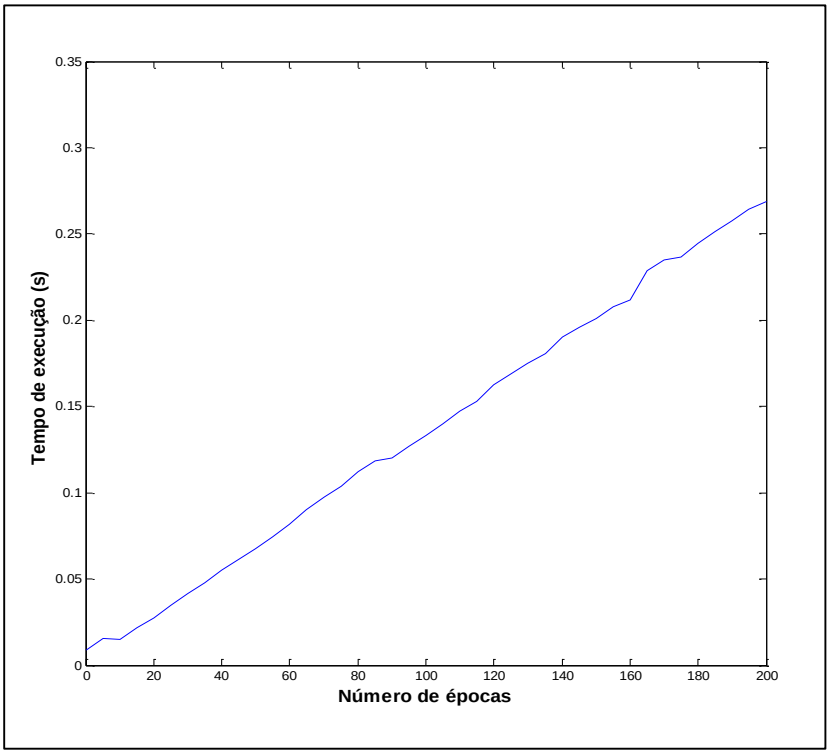

Figura 4.6 - Gráfico do tempo de execução do código por número de épocas para a rede RNA *Backpropagation* sem ruído utilizando na entrada dígitos de 7 segmentos.

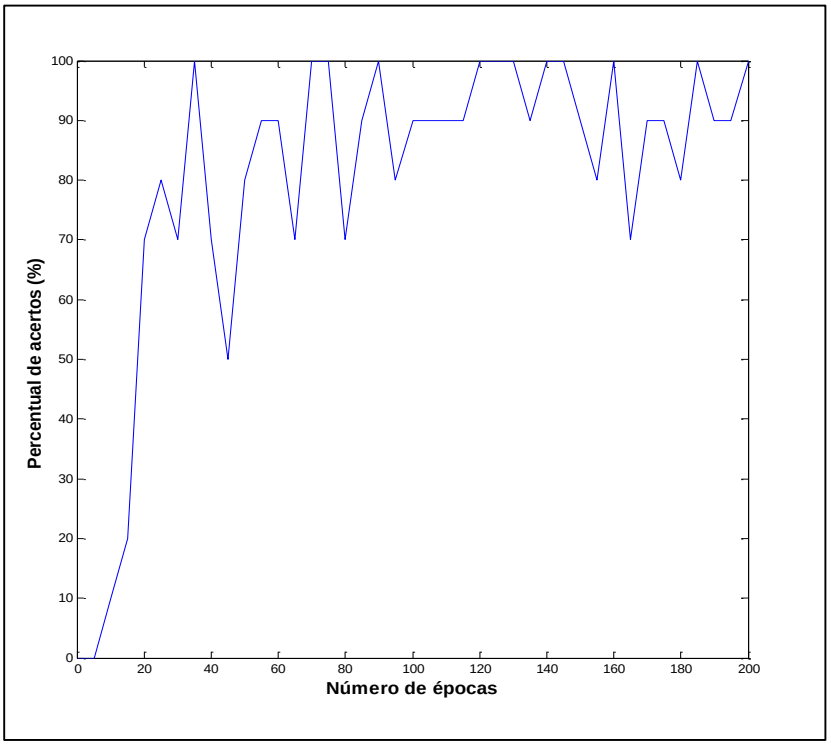

Figura 4.7 - Gráfico do percentual de acertos por número de épocas para a rede RNA *Backpropagation* com ruído utilizando na entrada dígitos de 7 segmentos.

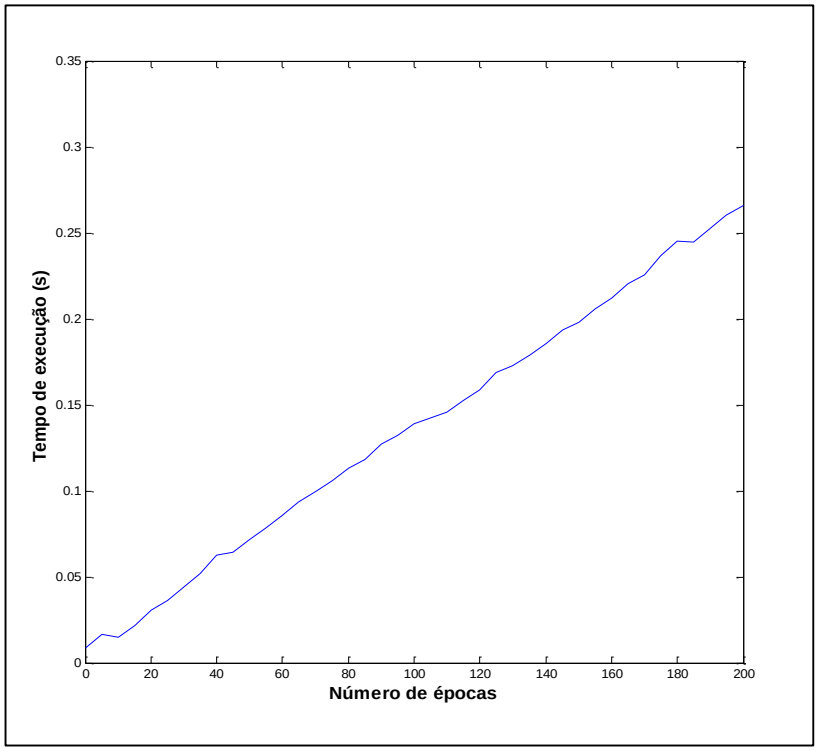

Figura 4.8 - Gráfico do tempo de execução do código por número de épocas para a rede RNA *Backpropagation* com ruído utilizando na entrada dígitos de 7 segmentos.

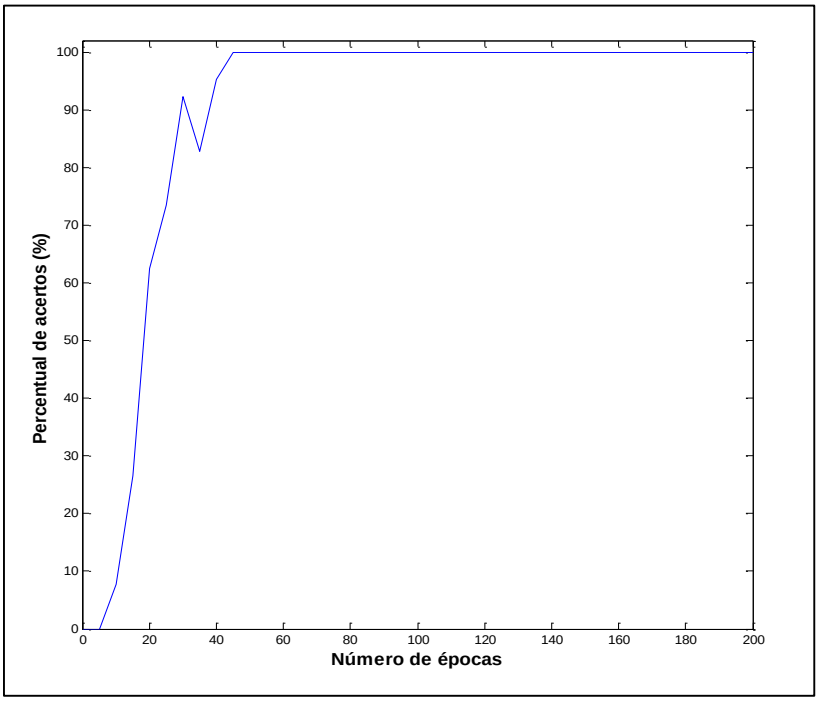

Figura 4.9 - Gráfico do percentual de acertos por número de épocas para a rede RNA *Backpropagation* sem ruído utilizando na entrada sequência binária de 6 *bits*.

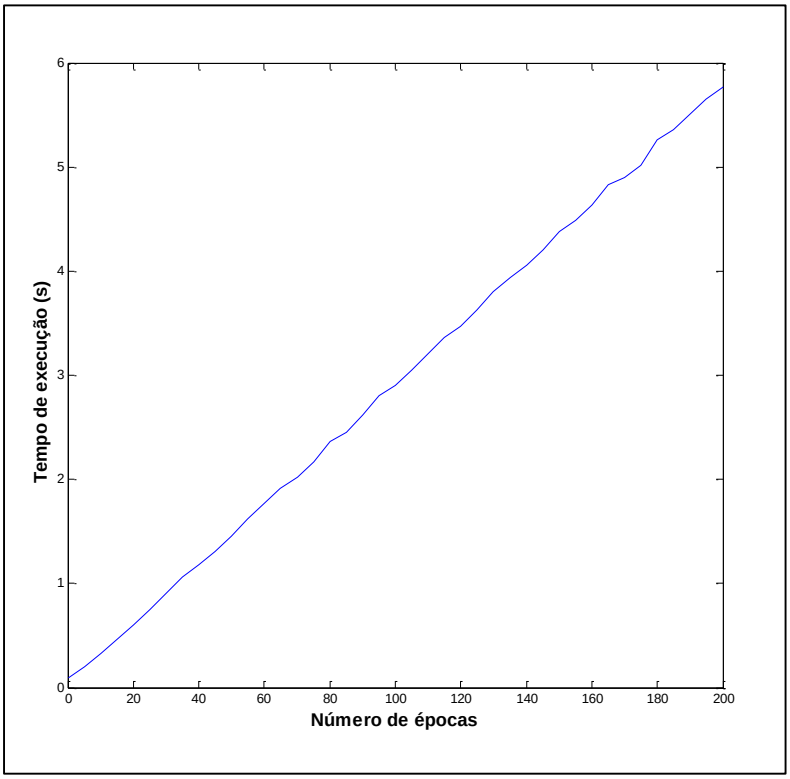

Figura 4.10 - Gráfico do tempo de execução do código por número de épocas para a rede RNA *Backpropagation* sem ruído utilizando na entrada sequência binária de 6 *bits*.

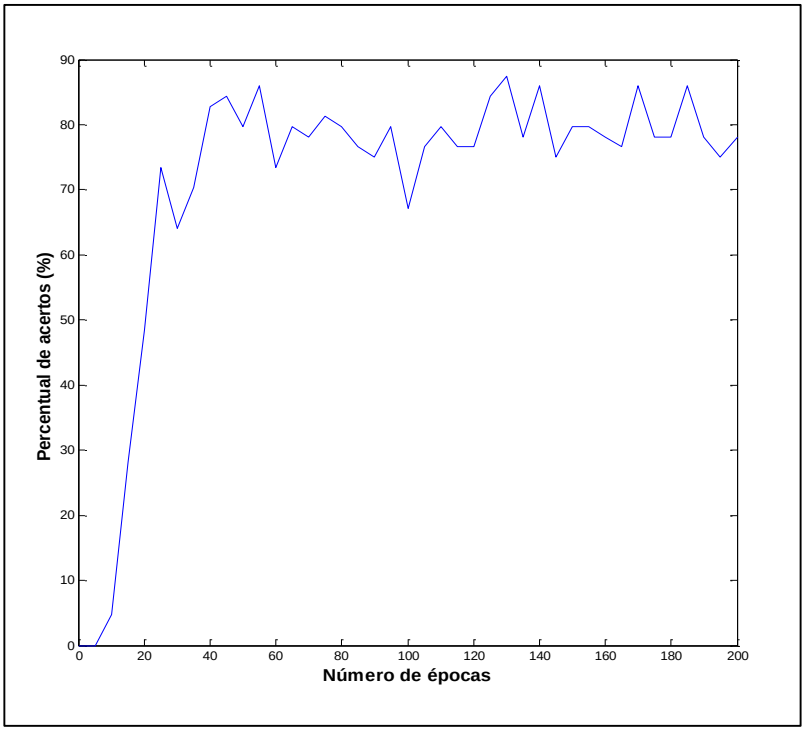

Figura 4.11 - Gráfico do percentual de acertos por número de épocas para a rede RNA *Backpropagation* com ruído utilizando na entrada sequência binária de 6 *bits*.

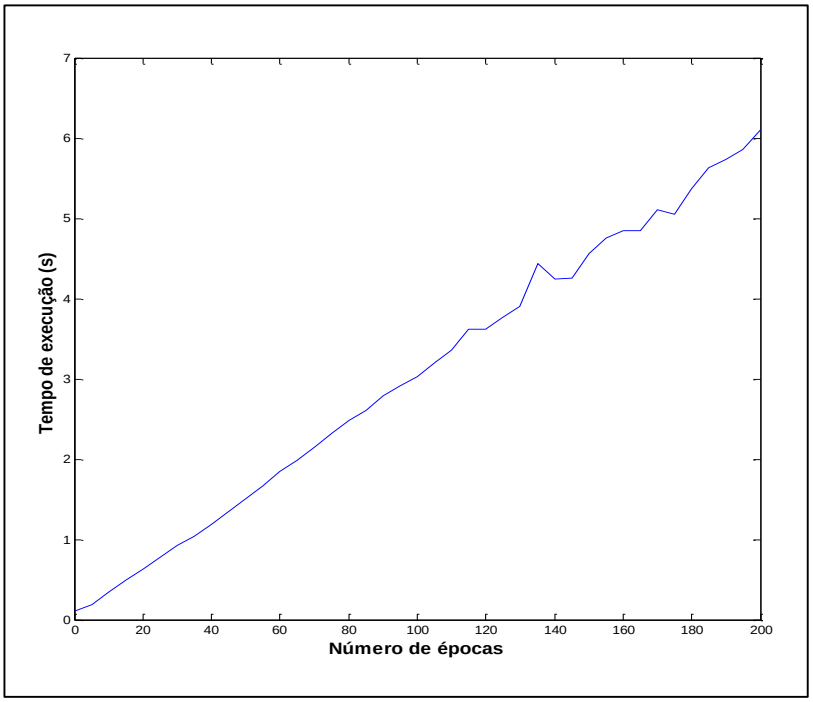

Figura 4.12 - Gráfico do tempo de execução do código por número de épocas para a rede RNA *Backpropagation* com ruído utilizando na entrada sequência binária de 6 *bits*.

Através das simulações foi possível constatar uma queda no percentual de reconhecimento da rede em condições com inserção de ruído. Nas simulações realizadas com uma quantidade maior de entrada, percebe-se que o percentual de reconhecimento de padrões manteve-se bom, que pode ser notado comparando as figuras 4.5 e 4.9. Lembrando que na figura 4.5 tem-se apenas 10 padrões de entrada enquanto que na figura 4.9 tem-se 64 padrões.

De qualquer forma, o algoritmo foi capaz de manter um bom percentual médio de acerto para os 64 padrões da sequência binária de 6 *bits*, na figura 4.9 este percentual ficou em torno de 100%. Já na figura 4.11, em que houve inserção de ruído, ocorreu uma queda no percentual de acerto de *bits*, o que era de se esperar.

Já o tempo de execução do código variou de acordo com a quantidade de épocas e dos padrões de entrada utilizados. Na figura 4.8, o tempo de execução do código foi 0,27 segundos para 200 épocas. Porém, na figura 4.12, o tempo de execução do código foi de aproximadamente 6 segundos para 200 épocas. Esta diferença se deve à quantidade maior de dados na entrada.

Estes gráficos gerados tiveram por objetivo traçar a evolução do algoritmo à medida que se aumenta o número de épocas. Através da tabela 4.1 a seguir é possível comparar os percentuais de acertos e tempos de execuções do código para as diferentes entradas, porém o

critério de parada utilizado na fase de treinamento foi de um erro médio quadrático de 0,001. Este erro é uma comparação do valor esperado e o valor encontrado na rede. As entradas utilizadas nesta tabela foram dígitos de 7 segmentos, sequência binária de 5 *bits* e sequência binária de 6 *bits*. Todas as simulações apresentaram um percentual de acerto de 100 % com um baixo tempo de execução. A tabela 4.2 apresenta as mesmas simulações, porém, com inserção de ruído, o que provocou uma queda no percentual de acertos.

Tabela 4.1 - Percentual de acertos do algoritmo RNA *Backpropagation* sem ruído.

Percentual de acertos para diferentes entradas sem ruído					
7 segmentos		Sequência binária de 5 *bits*		Sequência binária de 6 *bits*	
Tempo	% de acertos	Tempo	% de acertos	Tempo	% de acertos
0,06	100 %	0,30	100 %	1,67	100 %

Tabela 4.2 - Percentual de acertos do algoritmo RNA *Backpropagation* com ruído.

Percentual de acertos para diferentes entradas com ruído					
7 segmentos		Sequência binária de 5 *bits*		Sequência binária de 6 *bits*	
Tempo	% de acertos	Tempo	% de acertos	Tempo	% de acertos
0,06	77 %	0,36	100 %	1,84	80,25 %

Novas simulações foram realizadas utilizando o algoritmo RNA *Backpropagation*, porém aumentou-se bastante a quantidade de dados na entrada da rede. A tabela 4.3 a seguir apresenta um comparativo do percentual de acertos para sequência binária de 7 *bits* e sequência binário de 8 *bits*. Já tabela 4.4 apresenta as mesmas simulações porém com inserção de ruído na camada de entrada.

Tabela 4.3 - Percentual de acertos do algoritmo RNA *Backpropagation* sem ruído.

Percentual de acertos para diferentes entradas sem ruído			
Sequência binária de 7 *bits*		Sequência binária de 8 *bits*	
Tempo	% de acertos	Tempo	% de acertos
22 s	100 %	129 s	100 %

Tabela 4.4 - Percentual de acertos do algoritmo RNA *Backpropagation* com ruído.

Percentual de acertos para diferentes entradas com ruído			
Sequência binária de 7 *bits*		Sequência binária de 8 *bits*	
Tempo	% de acertos	Tempo	% de acertos
21 s	82 %	128 s	86 %

Portanto, através das tabelas 4.3 e 4.4 foi possível notar que a sequência binária de 8 *bits* apresentou um grande aumento no tempo de execução do código chegando a 129 segundos. Nota-se na tabela 4.4 que a inserção de ruído provocou uma significativa queda no percentual de acertos, diminuindo de 100 % para 86 % no caso da sequência binária de 8 *bits*.

4.5. SIMULAÇÕES COM MÁQUINA DE *BOLTZMANN* GERAL

As simulações realizadas com o algoritmo Máquina de *Boltzmann* Geral, utilizaram a estrutura de rede descrita na seção 3.2 e apresentada na figura 3.1.

Assim como nas simulações anteriores, foram realizados alguns testes variando a quantidade de padrões de entrada. Em um dos testes foi utilizado uma sequência binária de 6 *bits* totalizando $2^6 = 64$ padrões diferentes de entrada, sendo cada padrão com 6 valores diferentes. O treinamento da rede também foi feito de forma a igualar valores dos padrões da camada de entrada aos da camada de saída.

Primeiramente, testou-se como entrada no algoritmo Máquina de *Boltzmann*, os padrões de 7 segmentos, depois testou-se utilizando como entrada as sequências binárias de 5 *bits*, sequência binária de 6 *bits*, sequência binária de 7 *bits* e, por fim, utilizou-se uma sequência binária de 8 *bits*. Na sequência de gráficos das figuras 4.13 a 4.20 a seguir, são plotadas as

simulações realizadas utilizando diferentes condições de entrada variando a quantidade de épocas de 0 a 2000. Desta forma é plotado o gráfico do percentual de acerto por número de épocas e em sequência é plotado gráfico do tempo de execução do código por número de épocas. São simuladas as execuções dos códigos para as condições de entrada de 7 segmentos e sequência binária de 6 *bits*. Também são realizadas simulações com a mesma inserção de ruído que foi descrita na equação 4.1.

Através das legendas dos gráficos a seguir, é possível identificar as condições de entrada que foram utilizadas nas simulações.

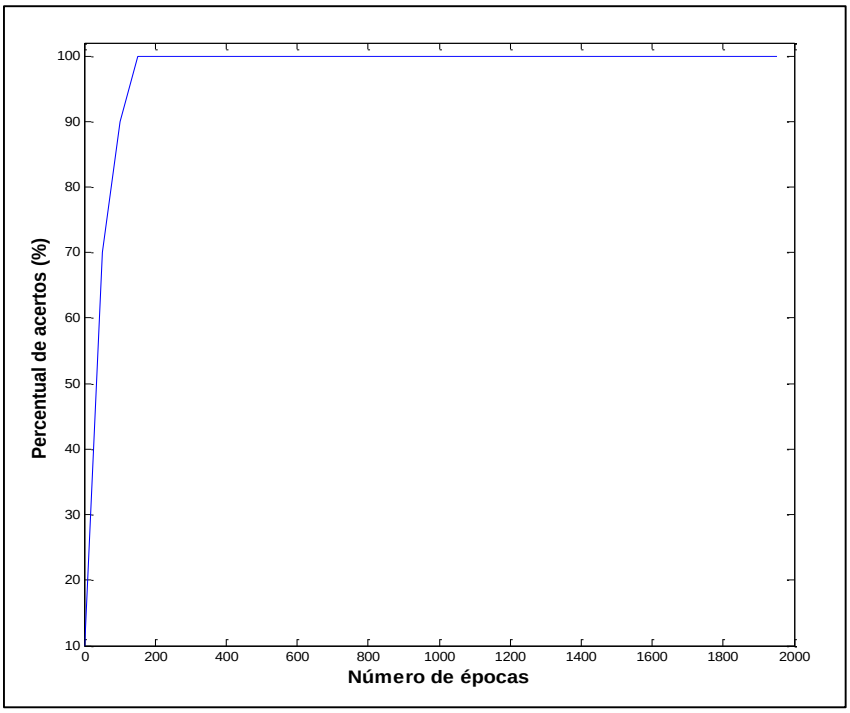

Figura 4.13 - Gráfico do percentual de acertos por número de épocas para a rede Máquina de *Boltzmann* Geral sem ruído utilizando na entrada dígitos de 7 segmentos.

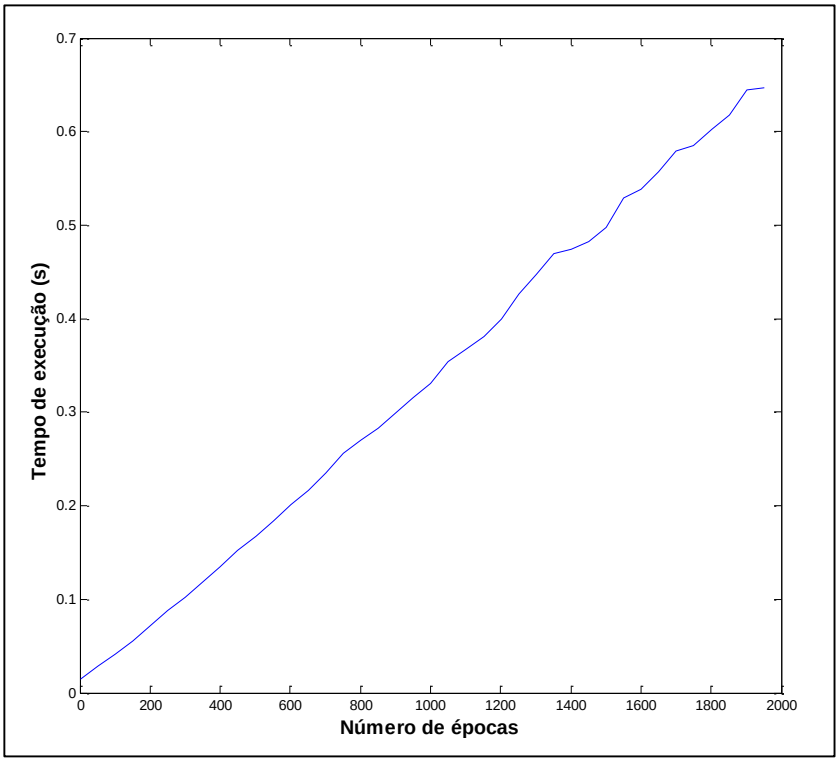

Figura 4.14 - Gráfico do tempo de execução do código por número de épocas para a rede Máquina de *Boltzmann* Geral sem ruído utilizando na entrada dígitos de 7 segmentos.

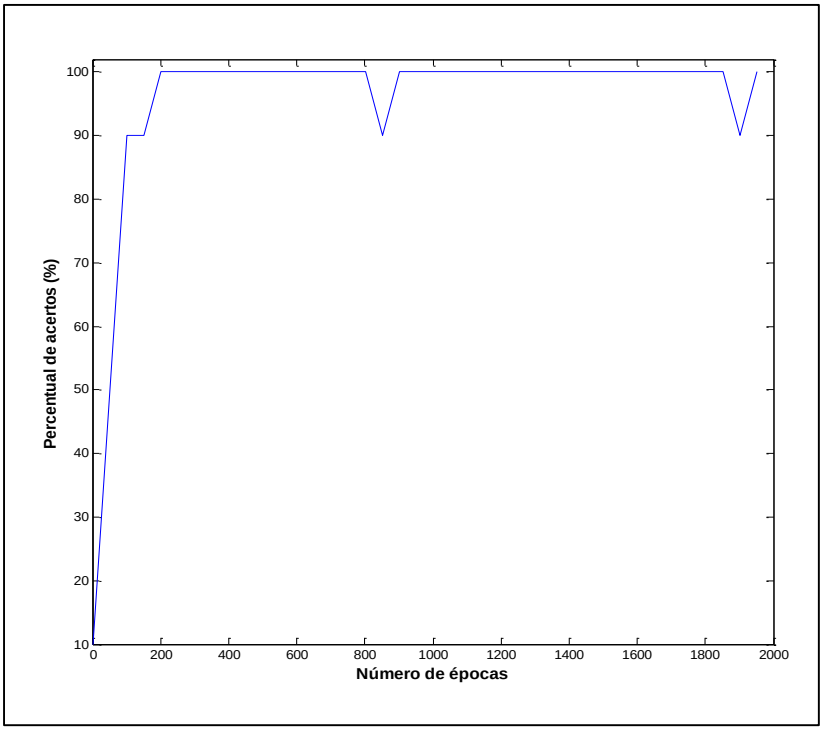

Figura 4.15 - Gráfico do percentual de acertos por número de épocas para a rede Máquina de *Boltzmann* Geral com ruído utilizando na entrada dígitos de 7 segmentos.

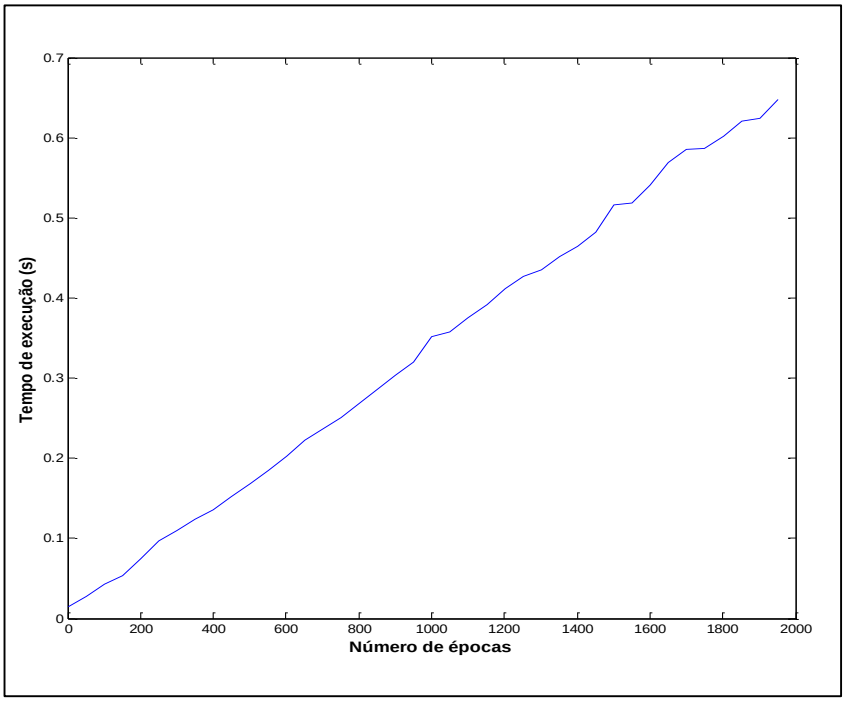

Figura 4.16 - Gráfico do tempo de execução do código por número de épocas para a rede Máquina de *Boltzmann* Geral com ruído utilizando na entrada dígitos de 7 segmentos.

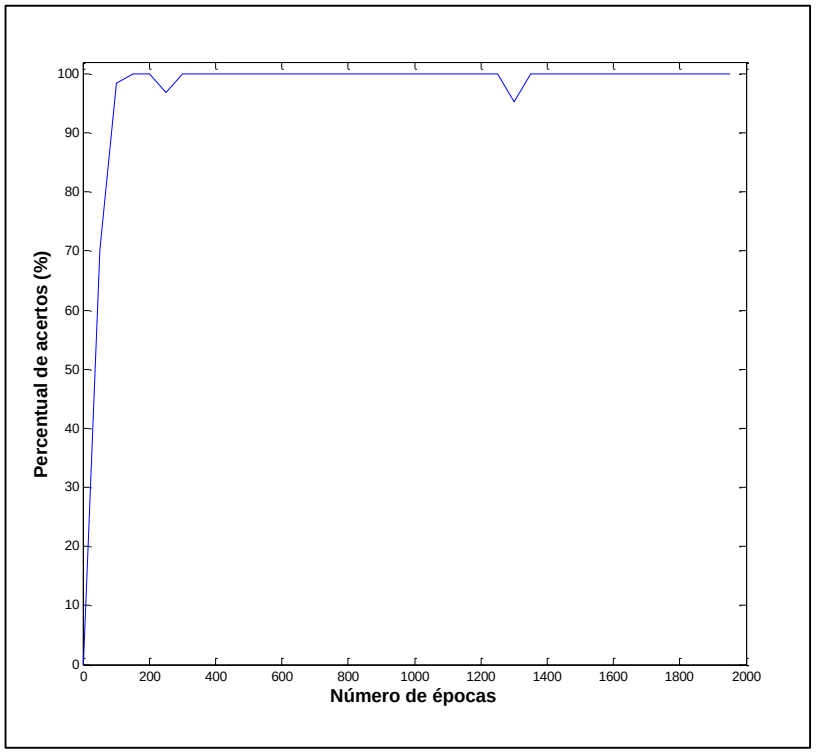

Figura 4.17 - Gráfico do percentual de acertos por número de épocas para a rede Máquina de *Boltzmann* Geral sem ruído utilizando na entrada sequência binária de 6 *bits*.

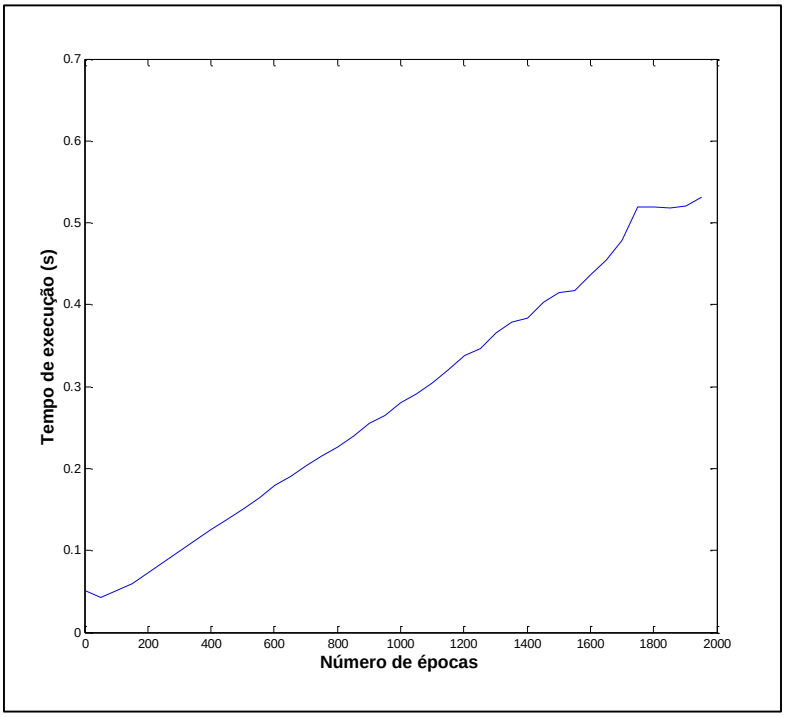

Figura 4.18 - Gráfico do tempo de execução do código por número de épocas para a rede Máquina de *Boltzmann* Geral sem ruído utilizando na entrada sequência binária de 6 *bits*.

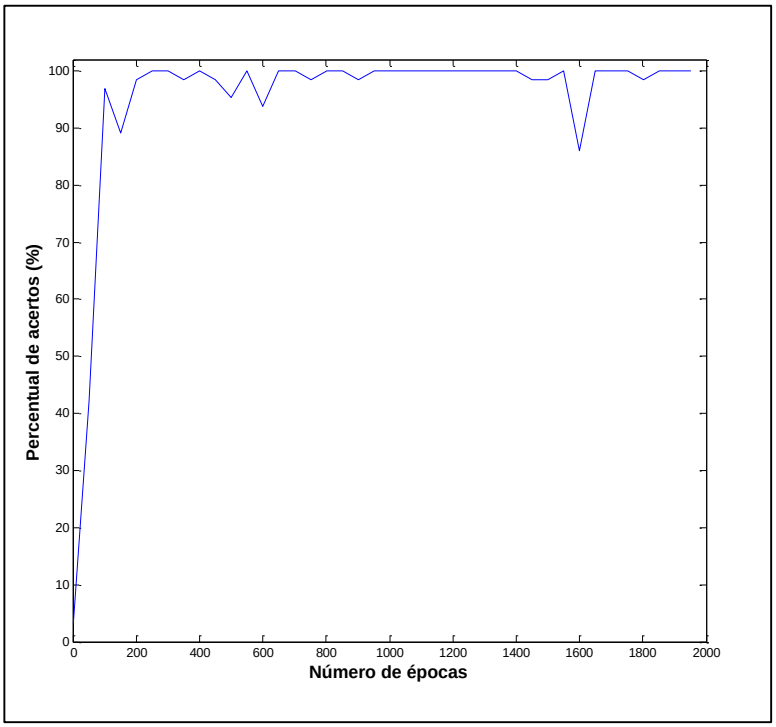

Figura 4.19 - Gráfico do percentual de acertos por número de épocas para a rede Máquina de *Boltzmann* Geral com ruído utilizando na entrada sequência binária de 6 *bits*.

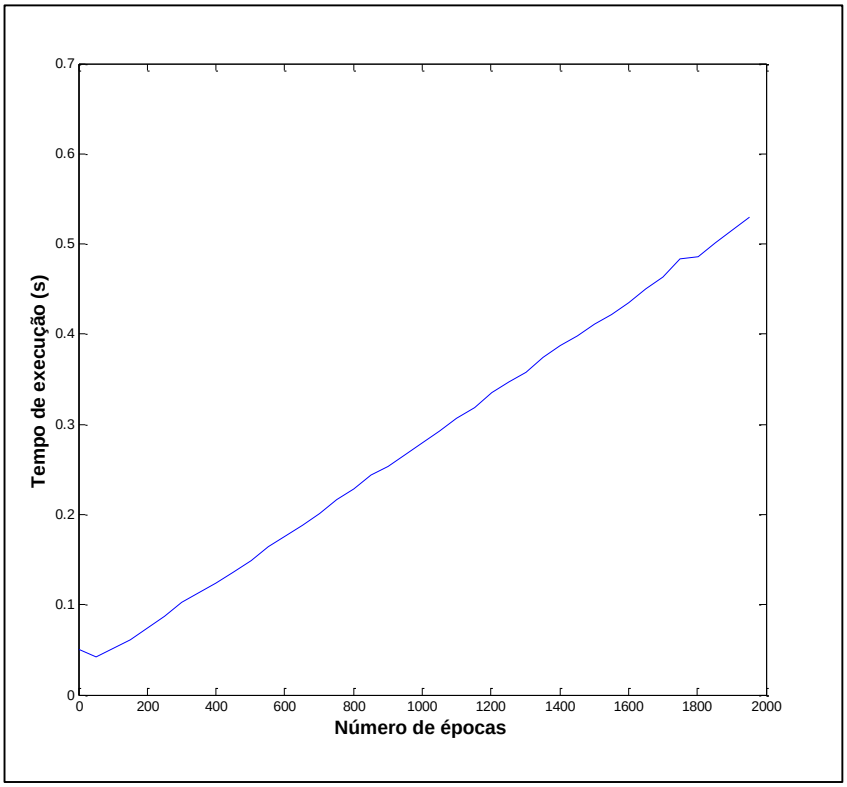

Figura 4.20 - Gráfico do tempo de execução do código por número de épocas para a rede Máquina de *Boltzmann* Geral com ruído utilizando na entrada sequência binária de 6 *bits*.

Nestas simulações realizadas foi possível constatar que o algoritmo Máquina de *Boltzmann* Geral apresentou ótimos percentuais de acertos. Nota-se que, nos gráficos gerados com ruído, o percentual de acerto de bits manteve-se muito bom.

Estes gráficos tiveram por objetivo avaliar a evolução do algoritmo Máquina de *Boltzmann* Geral de acordo a quantidade de épocas que o código é executado. Pode-se perceber claramente que com apenas 200 épocas de execução do código, o algoritmo já é capaz de apresentar ótimos resultados em percentuais de acertos.

Novas simulações foram realizadas utilizando o algoritmo Máquina de *Boltzmann* Geral, desta vez o número de épocas utilizadas variou de acordo com a quantidade de padrões a serem treinados. A quantidade de épocas escolhidas para execução do algoritmo foi uma quantidade suficiente para manter um percentual de acertos em torno de 100 % e foi determinada de forma experimental.

A tabela 4.5 a seguir apresenta o comparativo do percentual de acertos para distintas condições de entrada simuladas. Foram utilizados como entrada os dígitos de 7 segmentos, a sequência binária de 5 *bits* e a sequência binária de 6 *bits*. Em todos esses casos, apresentou-se

um percentual de acertos de 100 % e com um tempo de execução do código muito baixo. Conforme esta tabela mostra, iniciou-se os testes executando 200 épocas para entrada de 7 segmentos, 500 épocas para entrada da sequência binária de 5 *bits* e 1000 épocas para entrada binária de 6 *bits*.

Já a tabela 4.6 apresenta os mesmos testes realizados porém com inserção de ruído de 0,3 ou 30 % da potência do sinal conforme já utilizado em outras simulações neste capítulo. Com estas simulações, nota-se uma pequena queda no percentual de acerto de *bits* quando se compara as tabelas 4.5 e 4.6.

Tabela 4.5 - Percentual de acertos do algoritmo Máquina de *Boltzmann* Geral sem ruído.

Percentual de acertos para diferentes entradas sem ruído					
7 segmentos (200 épocas)		Sequência binária de 5 bits (500 épocas)		Sequência binária de 6 bits (1000 épocas)	
Tempo	% de acertos	Tempo	% de acertos	Tempo	% de acertos
0,05	100 %	0,11	100 %	0,28	100 %

Tabela 4.6 - Percentual de acertos do algoritmo Máquina de *Boltzmann* Geral com ruído.

Percentual de acertos para diferentes entradas com ruído (10 épocas)					
7 segmentos (200 épocas)		Sequência binária de 5 bits (500 épocas)		Sequência binária de 6 bits (1000 épocas)	
Tempo	% de acertos	Tempo	% de acertos	Tempo	% de acertos
0,05	92 %	0,11	100 %	0,29	98,88 %

Outras simulações foram realizadas para se testar a capacidade do algoritmo Máquina de *Boltzmann* Geral com a utilização de uma sequência binária de 7 *bits* e uma sequência binária de 8 *bits*. Foram executadas 1000 épocas para simulação da sequência de 7 *bits* e 2500 épocas para a sequência binária de 8 *bits*. Os resultados dessas simulações são apresentados na tabela 4.7 a seguir. Já na tabela 4.8 são apresentados os resultados com a inserção de um ruído de 0,3 na camada de entrada.

Tabela 4.7 - Percentual de acertos do algoritmo Máquina de *Boltzmann* Geral sem ruído.

Percentual de acertos para diferentes entradas sem ruído			
Sequência binária de 7 bits (1000 épocas)		Sequência binária de 8 bits (2500 épocas)	
Tempo	% de acertos	Tempo	% de acertos
0,38	99,52 %	1,18	98,2 %

Tabela 4.8 - Percentual de acertos do algoritmo Máquina de *Boltzmann* Geral com ruído.

Percentual de acertos para diferentes entradas com ruído			
Sequência binária de 7 bits (1000 épocas)		Sequência binária de 8 bits (2500 épocas)	
Tempo	% de acertos	Tempo	% de acertos
0,38	98,74 %	1,18	97 %

Através dessas simulações foi possível notar um grande percentual de acertos e um baixo de tempo de execução do código como nota-se nas tabelas 4.5, 4.6, 4.7 e 4.8.

Portanto, conclui-se através das tabelas descritas e dos gráficos mostrados, que a rede Máquina de *Boltzmann* Geral é capaz de realizar classificação de padrões com um tempo de execução inferior à RNA *Backpropagation* e com melhores resultados em termos de percentual de acertos.

4.6. SIMULAÇÕES COM MÁQUINA DE *BOLTZMANN* RESTRITA

Por fim, foi realizada simulações com o algoritmo Máquina de *Boltzmann* Restrita. A estrutura de rede utilizada foi descrita na seção 3.6 e demonstrada na figura 3.5.

Para comparar o desempenho desta rede em relação à rede RNA *Backpropagation* e Máquina *Boltzmann* Geral, a RBM utilizou como valores de entrada os mesmos padrões dos itens 4.2 e 4.3. Primeiramente, utilizou-se como entrada os padrões de 7 segmentos, a sequência binária de 5 *bits* e a sequência binária de 6 *bits*. Para cada um destes padrões de entrada, foram também acrescidos ruídos de 0,3 ou 30 % da potência do sinal conforme a equação 4.1.

Os gráficos de algumas simulações realizadas utilizando a rede RBM, são exibidos nos gráficos das figuras de 4.21 a 4.28 a seguir. Assim como nos algoritmos anteriores, são apresentados os gráficos de percentual de acerto por número de épocas para diferentes

condições de entrada, variando as épocas de 0 a 2000. Logo em seguida, é apresentado o gráfico do tempo de execução do código por número de épocas. Através das legendas destes gráficos, é possível identificar as condições de entradas utilizadas nas simulações.

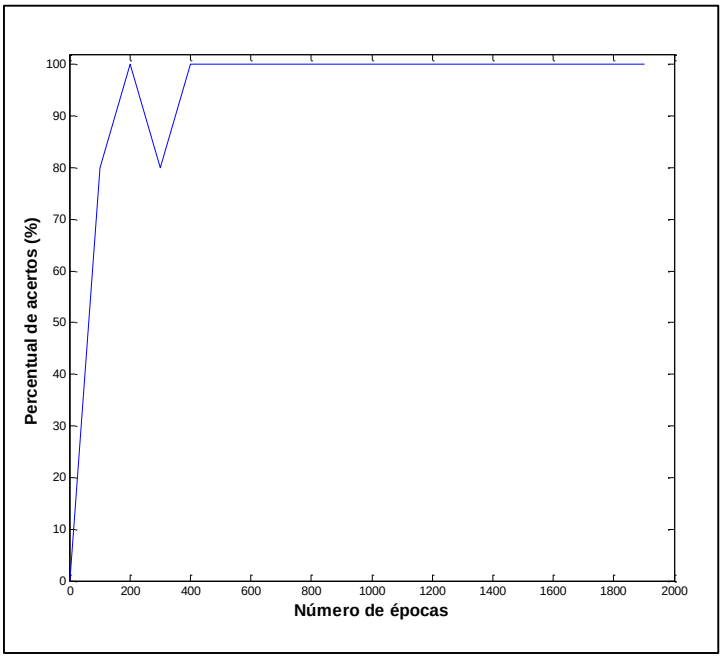

Figura 4.21 - Gráfico do percentual de acertos por número de épocas para a rede Máquina de *Boltzmann* Restrita sem ruído utilizando na entrada dígitos de 7 segmentos.

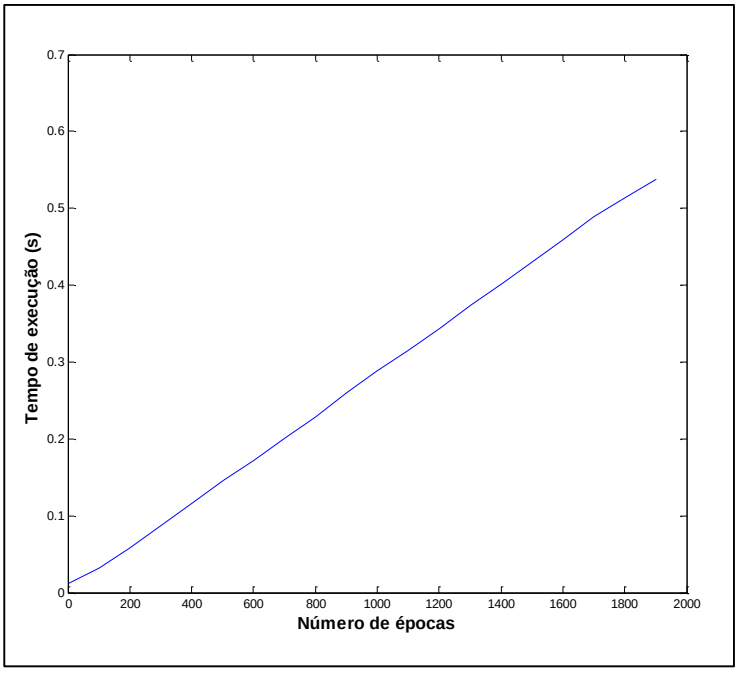

Figura 4.22 - Gráfico do tempo de execução do código por número de épocas para a rede Máquina de *Boltzmann* Restrita sem ruído utilizando na entrada dígitos de 7 segmentos.

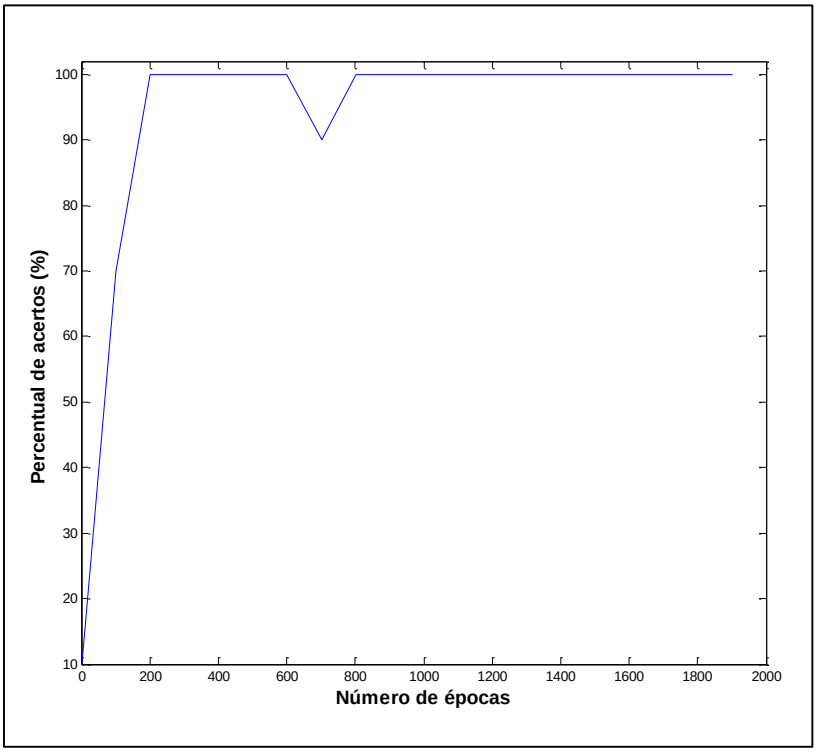

Figura 4.23 - Gráfico do percentual de acertos por número de épocas para a rede Máquina de *Boltzmann* Restrita com ruído utilizando na entrada dígitos de 7 segmentos.

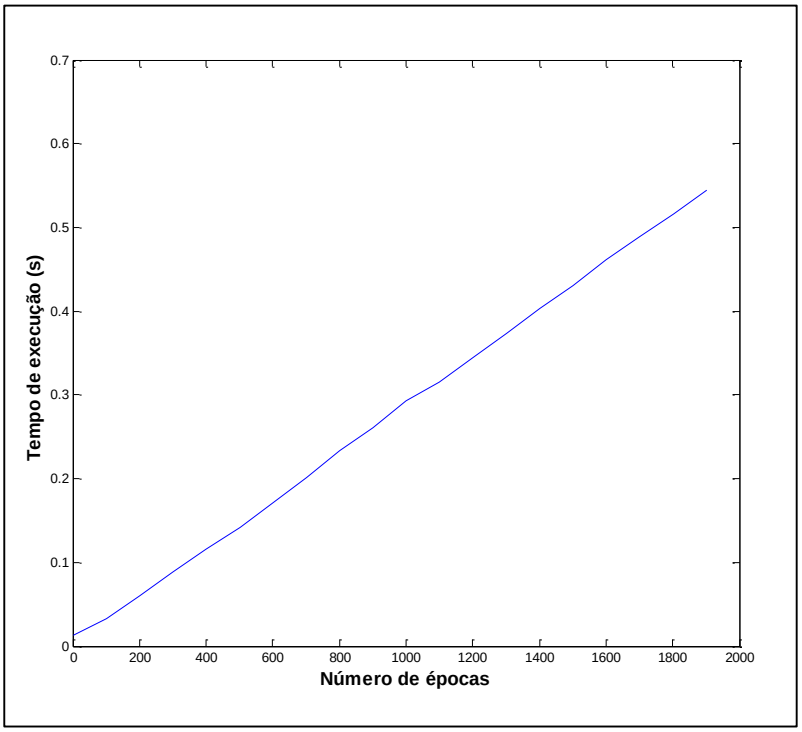

Figura 4.24 - Gráfico do tempo de execução do código por número de épocas para a rede Máquina de *Boltzmann* Restrita com ruído utilizando na entrada dígitos de 7 segmentos.

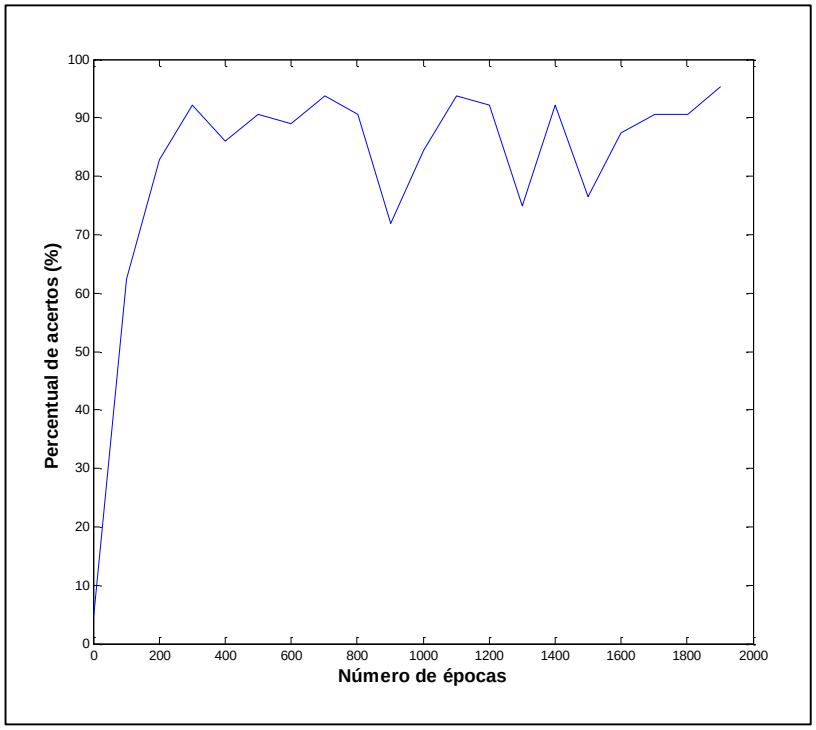

Figura 4.25 - Gráfico do percentual de acertos por número de épocas para a rede Máquina de *Boltzmann* Restrita sem ruído utilizando na entrada sequencia binária de 6 *bits*.

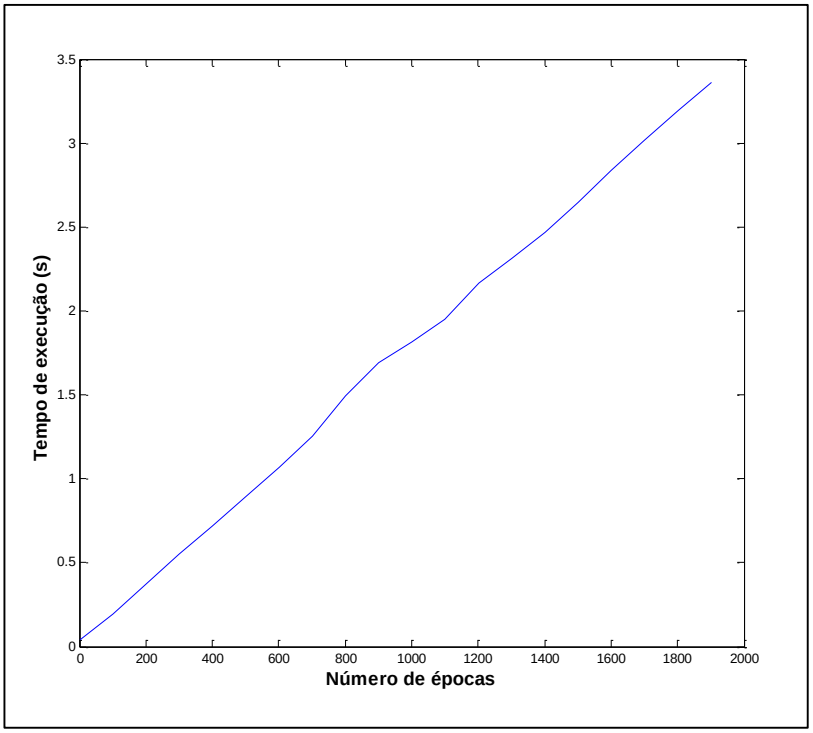

Figura 4.26 - Gráfico do tempo de execução do código por número de épocas para a rede Máquina de *Boltzmann* Restrita sem ruído utilizando na entrada sequencia binária de 6 *bits*.

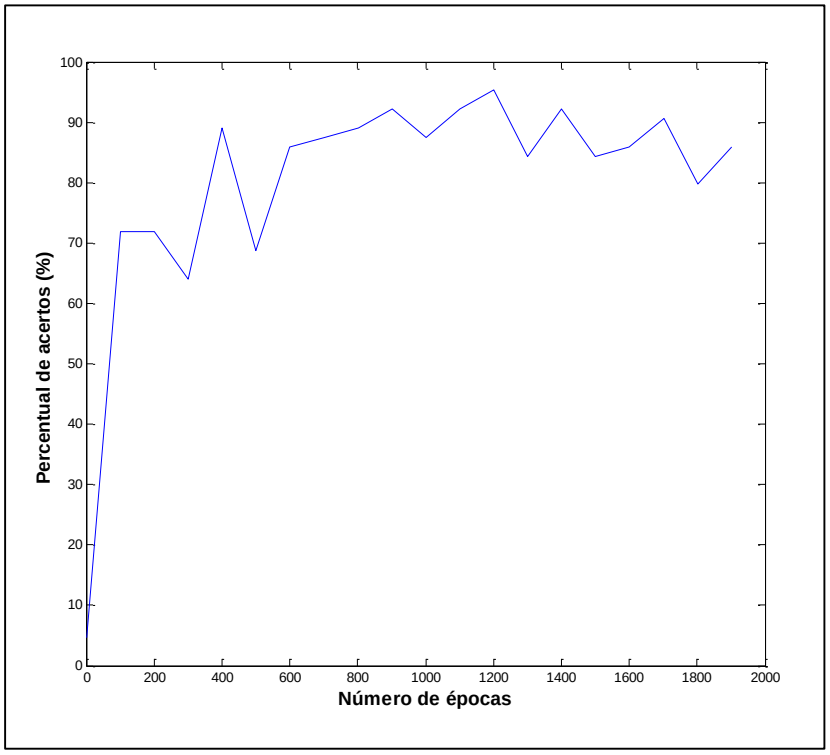

Figura 4.27 - Gráfico do percentual de acertos por número de épocas para a rede Máquina de *Boltzmann* Restrita com ruído utilizando na entrada sequencia binária de 6 *bits*.

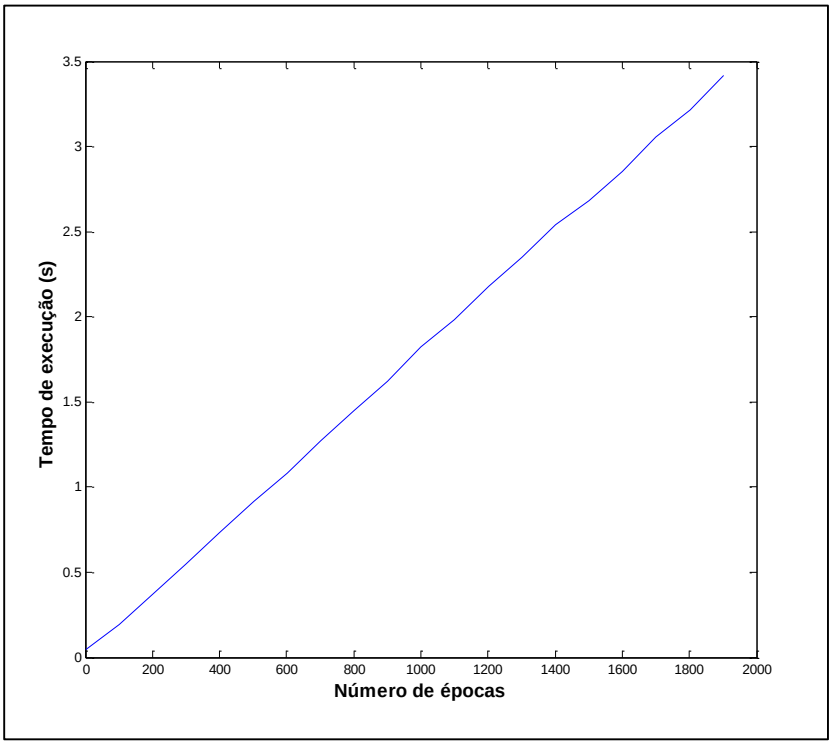

Figura 4.28 - Gráfico do tempo de execução do código por número de épocas para a rede Máquina de *Boltzmann* Restrita com ruído utilizando na entrada sequencia binária de 6 *bits*.

Através dos gráficos gerados com o algoritmo RBM, foi possível notar um bom percentual de acertos para condições com maior número de épocas. As plotagens dos gráficos tiveram o objetivo de avaliar a evolução do percentual de acertos de acordo com o número de épocas. Foi possível constatar que quando se aumenta o número de padrões na camada de entrada da rede, são necessárias várias épocas neste algoritmo para se conseguir um bom percentual de acertos.

Novas simulações foram realizadas com o algoritmo Máquina de *Boltzmann* Restrita e desta vez foi fixada a quantidade de épocas em 1500. Através da tabela 4.9 a seguir é apresentado os resultados dos percentuais de acertos de *bits* e tempo de execução do código para diferentes condições de entrada como: dígitos de 7 segmentos, sequência binária de 5 *bits* e sequência binária de 6 *bits*.

Já a tabela 4.10 apresenta os resultados para as mesmas simulações porém com a inserção de um ruído de 0,3 ou 30 % da potência do sinal na camada de entrada. Nota-se nestas simulações que o percentual de acerto de *bits* manteve-se muito bom e quando se inseriu ruído houve uma queda no percentual de acertos.

Tabela 4.9 - Percentual de acertos do algoritmo Máquina de *Boltzmann* Restrita.

Percentual de acertos para diferentes entradas sem ruído (1500 épocas)					
7 segmentos		Sequência binária de 5 *bits*		Sequência binária de 6 *bits*	
Tempo	% de acertos	Tempo	% de acertos	Tempo	% de acertos
0,52	100 %	1,80	97 %	3,34	96 %

Tabela 4.10 - Percentual de acertos do algoritmo Máquina de *Boltzmann* Restrita.

Percentual de acertos para diferentes entradas com ruído (1500 épocas)					
7 segmentos		Sequência binária de 5 *bits*		Sequência binária de 6 *bits*	
Tempo	% de acertos	Tempo	% de acertos	Tempo	% de acertos
0,52	100 %	1,60	95 %	3,30	94 %

Outras simulações foram realizadas com o algoritmo Máquina de *Boltzmann* Restrita porém com 2000 épocas de execução, foi necessário aumentar o número de épocas para conseguir um melhor percentual de acerto de *bits*. Através da tabela 4.11 são apresentados os valores obtidos através da simulação deste algoritmo porém utilizando na entrada uma sequência binária de 7 *bits* e uma sequência binária de 8 *bits*. Na tabela 4.12 são apresentadas as mesmas simulações com inserção de ruído de 0,3, notando assim uma queda no percentual de acerto de *bits* em relação a simulação anterior.

Tabela 4.11 - Percentual de acertos do algoritmo Máquina de *Boltzmann* Restrita.

Percentual de acertos para diferentes entradas sem ruído (2000 épocas)			
Sequência binária de 7 *bits*		Sequência binária de 8 *bits*	
Tempo	% de acertos	Tempo	% de acertos
7,41 s	93,70 %	14,5 s	91 %

Tabela 4.12 - Percentual de acertos do algoritmo Máquina de *Boltzmann* Restrita.

Percentual de acertos para diferentes entradas com ruído (2000 épocas)			
Sequência binária de 7 *bits*		Sequência binária de 8 *bits*	
Tempo	% de acertos	Tempo	% de acertos
7,3 s	92 %	14,6 s	88 %

Portanto, conclui-se que a rede Máquina de *Boltzmann* Restrita também é capaz de realizar classificação de padrões com um tempo de execução inferior à RNA *Backpropagation* e com um percentual de acertos inferior ao algoritmo Máquina de *Boltzmann* Geral.

4.7. COMPARATIVO DOS PERCENTUAIS DE ACERTOS DOS 3 ALGORITMOS SIMULADOS

Através das simulações realizadas neste capítulo foi possível gerar a tabela 4.13 que apresenta um comparativo do percentual de acertos de *bits* dos 3 algoritmos simulados ao longo deste capítulo, utilizando na entrada uma sequência binária de 6 *bits*. Já a tabela 4.14 apresenta

os resultados com a sequência binária de 6 *bits* porém com inserção de ruído nos dados de entrada.

Analisando os valores nestas tabelas foi possível notar que o algoritmo Máquina de *Boltzmann* Geral apresentou o menor tempo de execução do código e o melhor percentual de acerto de *bits* para as condições com ruído e sem ruído. Nota-se também que algoritmo RNA *Backpropagation* quando se insere ruído apresenta os piores resultados pois há maior queda no percentual de acertos. Já o algoritmo Máquina de *Boltzmann* Restrita apresentou um tempo de execução do código um pouco maior do que o do RNA *Backpropagation* porém com um melhor percentual de acerto de *bits* quando se tem ruído nos dados de entrada.

Tabela 4.13 - Comparativo do percentual de acertos para 3 algoritmos simulados sem ruído.

Percentual de acertos para sequência binária de 6 *bits* na entrada					
RNA *Backpropagation*		Máquina de *Boltzmann* Geral		Máquina *Boltzmann* Restrita	
Tempo	% de acertos	Tempo	% de acertos	Tempo	% de acertos
1,67 s	100 %	0,28 s	100 %	3,34 s	96 %

Tabela 4.14 - Comparativo do percentual de acertos para 3 algoritmos simulados com ruído.

Percentual de acertos para sequência binária de 6 *bits* na entrada					
RNA *Backpropagation*		Máquina de *Boltzmann* Geral		Máquina *Boltzmann* Restrita	
Tempo	% de acertos	Tempo	% de acertos	Tempo	% de acertos
1,84 s	80,25 %	0,29 s	98,88 %	3,30 s	94 %

A tabela 4.15 a seguir apresenta um comparativo dos percentuais de acertos de *bits* e tempo de execução do código para a sequência binária de 7 *bits* na camada de entrada, para os 3 algoritmos simulados nesta pesquisa. Já a tabela 4.16 compara os resultados quando se insere ruído nos dados de entrada.

Através das comparações destas tabelas, é possível notar que algoritmo Máquina de *Boltzmann* Geral apresentou melhores percentuais de acertos e com baixo tempo de execução do código comparado aos demais algoritmos. Já o algoritmo RNA *Backpropagation* apresentou

um bom percentual de acertos de *bits* porém com um grande tempo de execução do código, chegando a 22 segundos enquanto o algoritmo RBM demorou 7,41 segundos e o algoritmo Máquina de Boltzmann Geral demorou apenas 0,38 segundos.

Quando se insere ruído conforme nota-se na tabela 4.16, o percentual de acerto de *bits* do algoritmo RNA *Backpropagation* novamente teve a maior queda no percentual de acertos comparada aos demais algoritmos. Já o algoritmo Máquina de *Boltzmann* Restrita apresentou percentual de acertos de *bits* superior ao RNA *Backpropagation* quando se compara a condição de uma entrada com ruído.

Tabela 4.15 - Comparativo do percentual de acertos para 3 algoritmos simulados sem ruído.

Percentual de acertos para sequência binária de 7 *bits* na entrada					
RNA *Backpropagation*		Máquina de *Boltzmann* Geral		Máquina *Boltzmann* Restrita	
Tempo	% de acertos	Tempo	% de acertos	Tempo	% de acertos
22 s	100 %	0,38 s	99,82 %	7,41 s	93,70 %

Tabela 4.16 - Comparativo do percentual de acertos para 3 algoritmos simulados com ruído.

Percentual de acertos para sequência binária de 7 *bits* na entrada					
RNA *Backpropagation*		Máquina de *Boltzmann* Geral		Máquina *Boltzmann* Restrita	
Tempo	% de acertos	Tempo	% de acertos	Tempo	% de acertos
21 s	82 %	0,38 s	98,74 %	7,3	92 %

Por fim, é apresentada a tabela 4.17 que apresenta os mesmos comparativos dos 3 algoritmos simulados, porém a entrada de dados comparada foi uma sequência binária de 8 *bits*. Na tabela 4.18 as mesmas comparações são realizadas porém com inserção de ruído na camada de entrada.

Através destas tabelas, novamente foi possível notar que o algoritmo Máquina de *Boltzmann* Geral apresentou melhores resultados com um baixo tempo de execução do código. O algoritmo RNA *Backpropagation* teve um tempo de execução muito superior aos demais

códigos chegando a 128 segundos, contra 1,18 segundos do algoritmo Máquina de *Boltzmann* Geral e 14,5 segundos do algoritmo Máquina de *Boltzmann* Restrita.

Tabela 4.17 - Comparativo do percentual de acertos para 3 algoritmos simulados sem ruído.

Percentual de acertos para sequência binária de 8 *bits* na entrada					
RNA *Backpropagation*		Máquina de *Boltzmann* Geral		Máquina *Boltzmann* Restrita	
Tempo	% de acertos	Tempo	% de acertos	Tempo	% de acertos
129 s	100 %	1,18 s	98,2 %	14,5 s	91 %

Tabela 4.18 - Comparativo do percentual de acertos para 3 algoritmos simulados com ruído.

Percentual de acertos para sequência binária de 8 bits na entrada					
RNA *Backpropagation*		Máquina de *Boltzmann* Geral		Máquina *Boltzmann* Restrita	
Tempo	% de acertos	Tempo	% de acertos	Tempo	% de acertos
128 s	86 %	1,18 s	97 %	14,6 s	88 %

Através dos gráficos gerados neste capítulo e das tabelas, foi possível notar a eficácia dos algoritmos Máquina de *Boltzmann* Geral e Máquina de *Boltzmann* Restrita em realizar classificação de padrões. Todas as análises foram feitas comparando estes 2 algoritmos com o algoritmo RNA *Backpropagation*. Os tempos de execuções dos algoritmos Máquina de *Boltzmann* Geral e RBM foram bem melhores que do algoritmo RNA *Backpropagation*. Além disso, quando se insere ruídos nos dados de entrada da rede, os algoritmos RBM e Maquina de *Boltzmann* Geral tiveram melhores percentuais de acerto de *bits* comparado ao RNA *Backpropagation*.

Por fim, notou-se claramente que o algoritmo Máquina de *Boltzmann* Geral apresentou os melhores resultados entre os 3 algoritmos simulados.

4.8. CONSIDERAÇÕES FINAIS DESTE CAPÍTULO

Neste capítulo foram realizadas simulações de três algoritmos diferentes de reconhecimento de padrões: RNA *Backpropagation*, Máquina de *Boltzmann* Geral e Máquina de *Boltzmann* Restrita. Para cada um dos algoritmos testados, foi utilizado como entrada diferentes padrões de entrada. Primeiramente foi utilizado 10 padrões de um *display* de 7 segmentos, depois foram utilizadas sequências binárias de 5 *bits*, 6 *bits*, 7 *bits* e 8 *bits*.

Através das simulações realizadas, pôde-se comparar os percentuais de acertos dos padrões após o treinamento da rede e o tempo de execução dos códigos. Ao final das simulações de cada algoritmo, notou-se que alguns algoritmos apresentam melhores resultados em relação ao percentual de acertos e alguns apresentaram maiores tempos de execução do código.

Conclui-se com estas comparações, que o algoritmo Máquina de *Boltzmann* Geral apresentou a melhor relação percentual de acertos e tempo de execução. A rede RNA *Backpropagation* com ruído apresentou uma maior queda no percentual de acerto de *bits* e possui um tempo de execução do código muito alto. Já a rede Máquina de *Boltzmann* Restrita possui também um baixo tempo de execução do código e bom percentual de acerto de *bits* porém inferior ao algoritmo Máquina de *Boltzmann* Geral.

Desta forma, os algoritmos Máquina de *Boltzmann* apresentaram ótimos resultados que podem ser aplicados para solucionar problemas de reconstrução e classificação de padrões.

CAPÍTULO 5

5. RECONSTRUÇÃO E CLASSIFICAÇÃO DE IMAGENS BINÁRIAS UTILIZANDO O ALGORITMO MÁQUINA DE *BOLTZMANN* RESTRITA

5.1. INTRODUÇÃO

Neste capítulo será abordada a aplicação do Algoritmo Máquina de *Boltzmann* Restrita para reconstrução e classificação de imagens. Diante dos testes realizados no capítulo anterior, os algoritmos Máquina de *Boltzmann* Geral e Máquina de *Boltzmann* Restrita apresentaram melhores resultados comparado ao algoritmo RNA *Backpropagation*.

Através da utilização de imagens, é possível utilizar uma quantidade maior de dados comparada à sequência binária do capítulo anterior. Também é possível visualizar uma utilização mais prática deste algoritmo pois existe uma grande aplicabilidade em problemas de reconhecimento de imagens.

Para testar a capacidade deste algoritmo em classificar padrões, foram utilizadas 3 diferentes imagens com dimensões de 256 por 256 *bits*. As imagens escolhidas são bem conhecidas e utilizadas em pesquisas da área de processamento digital de imagens, como: a imagem da Lenna conforme figura 5.1, a imagem de uma ponte na figura 5.2 e imagem de um Câmera exibida na figura 5.3.

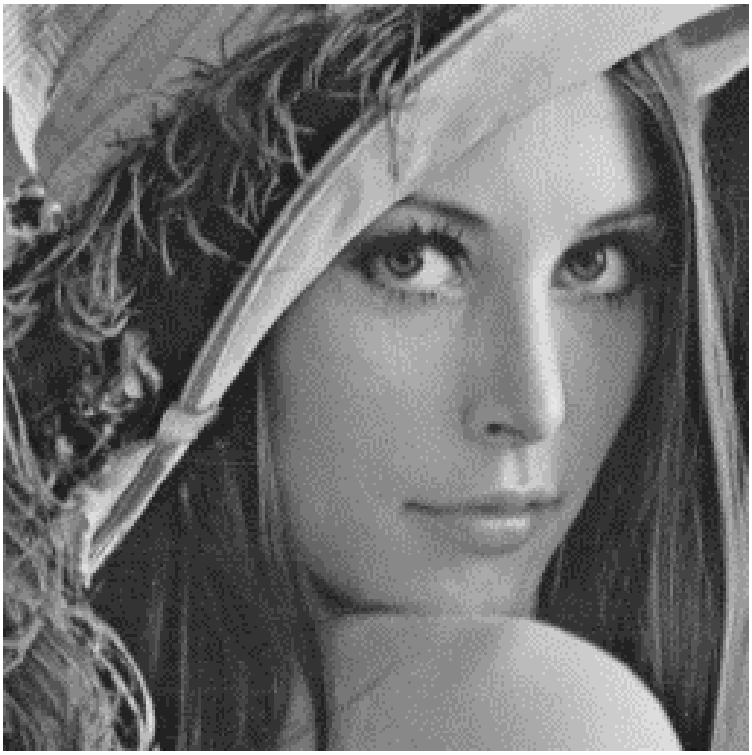

Figura 5.1 - Imagem 1: Lenna.

Figura 5.2 - Imagem 2: Ponte.

Figura 5.3 - Imagem 3: Camera.

Para realizar os testes no algoritmo RBM, as imagens foram identificadas conforme a figura 5.4 com numerações como: Imagem 1, Imagem 2 e Imagem 3. Nas simulações realizadas, a fase de treinamento selecionou uma destas imagens, e na fase de reconstrução foi utilizado os pesos treinados das mesmas imagens e em alguns casos os pesos de outra imagem. Esta variação de utilização de pesos na fase de reconstrução, teve por objetivo testar a eficácia do algoritmo em realmente reconstruir padrões.

Figura 5.4 - Identificação numérica das imagens treinadas para reconstrução.

5.2. RECONSTRUÇÃO DE IMAGENS UTILIZANDO O ALGORITMO RBM

Inicialmente, a imagem selecionada para teste foi binarizada. Como os pixels das imagens possuem valores variando de 0 a 255, foi definido o valor "0" para os pixels com valores entre 0 e 127 e o valor "1" para os pixels com valores entre 128 e 255. Após este processo de binarização, a primeira imagem testada pode ser visualizada pela figura 5.5, conforme nota-se a imagem da Lenna.

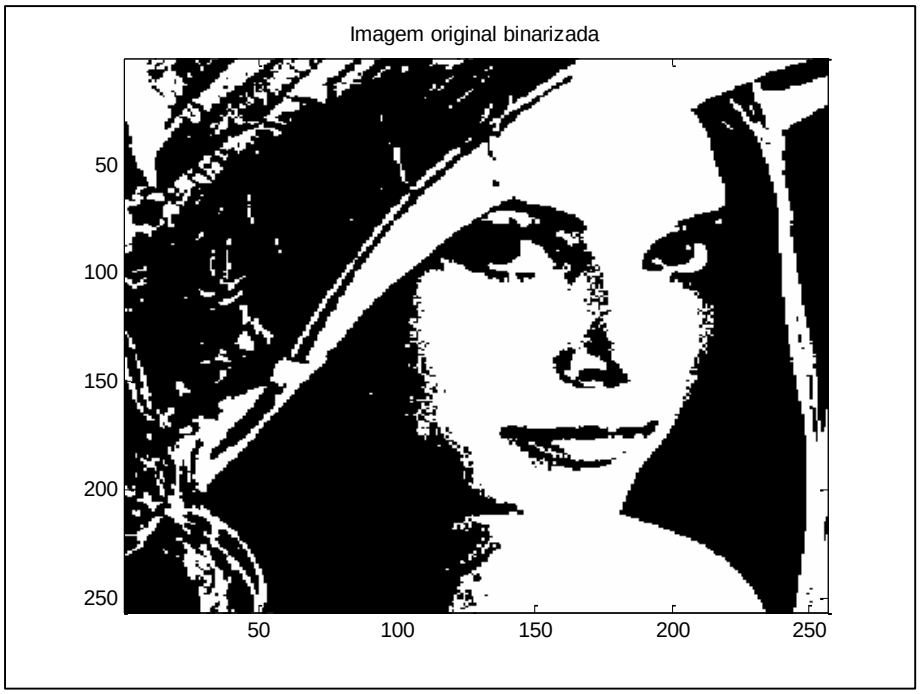

Figura 5.5 - Imagem da Lenna binarizada.

Além disso, foi introduzido um ruído aleatório na imagem através da aplicação da equação 5.1 no algoritmo RBM, conforme a figura 5.6 a seguir.

$$v(m) = x(imagem,n,m)+1.5*(rand-.5) \quad (5.1)$$

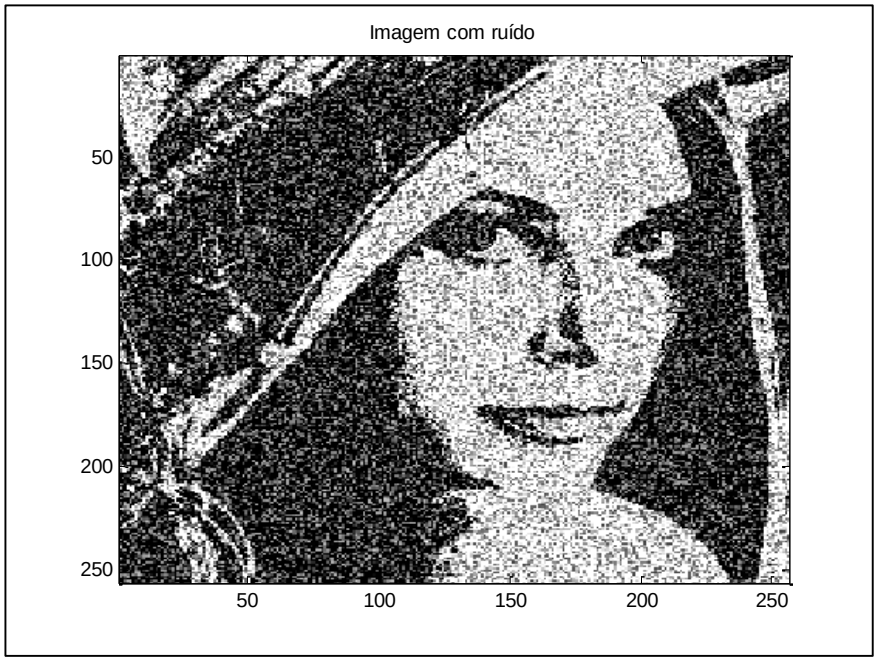

Figura 5.6 - Imagem da Lenna com inserção de ruído aleatório.

A inserção deste ruído na imagem, teve por objetivo testar a capacidade da rede RBM em reconstruir uma imagem ruidosa. Na figura 5.7 é apresentada a reconstrução da imagem após o treinamento dos pesos pelo algoritmo RBM. Para este teste houve a execução de 20 épocas de treinamento e o percentual de acertos dos bits foi de 87,25 %.

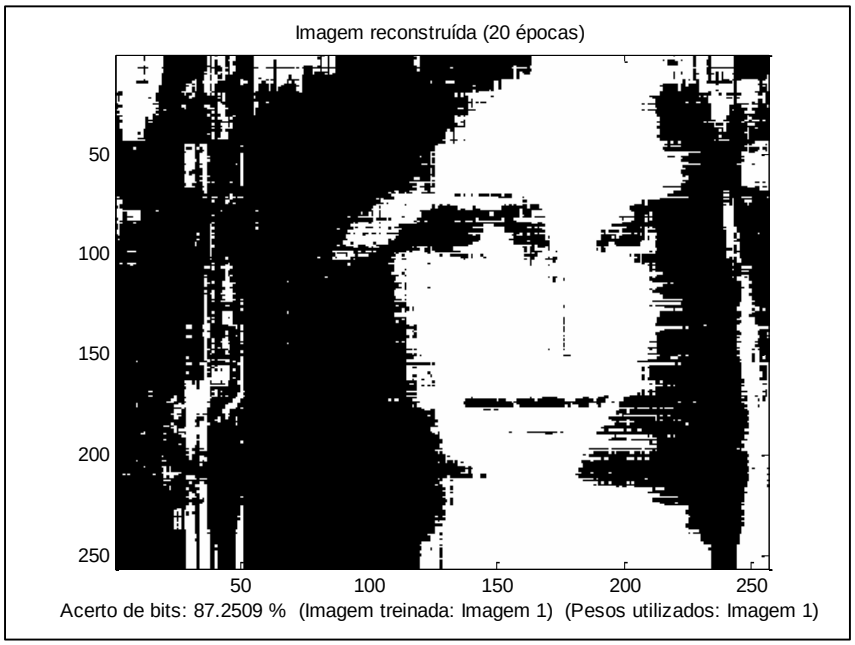

Figura 5.7 - Imagem da Lenna reconstruída com 20 épocas a partir dos pesos da Imagem 1.

Foi realizado um novo treinamento conforme a figura 5.8, desta vez com 40 épocas e o percentual de acertos de 94,65 %. Já na figura 5.9 o treinamento realizado foi de 60 épocas e percentual de acerto total de bits foi 96,68 %.

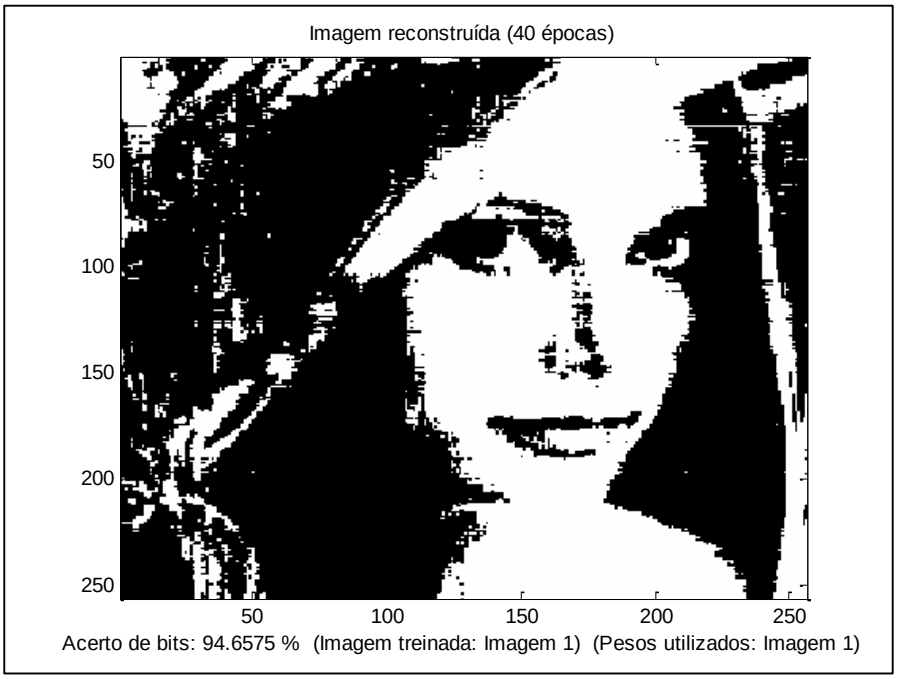

Figura 5.8 - Imagem da Lenna reconstruída com 40 épocas a partir dos pesos da Imagem 1.

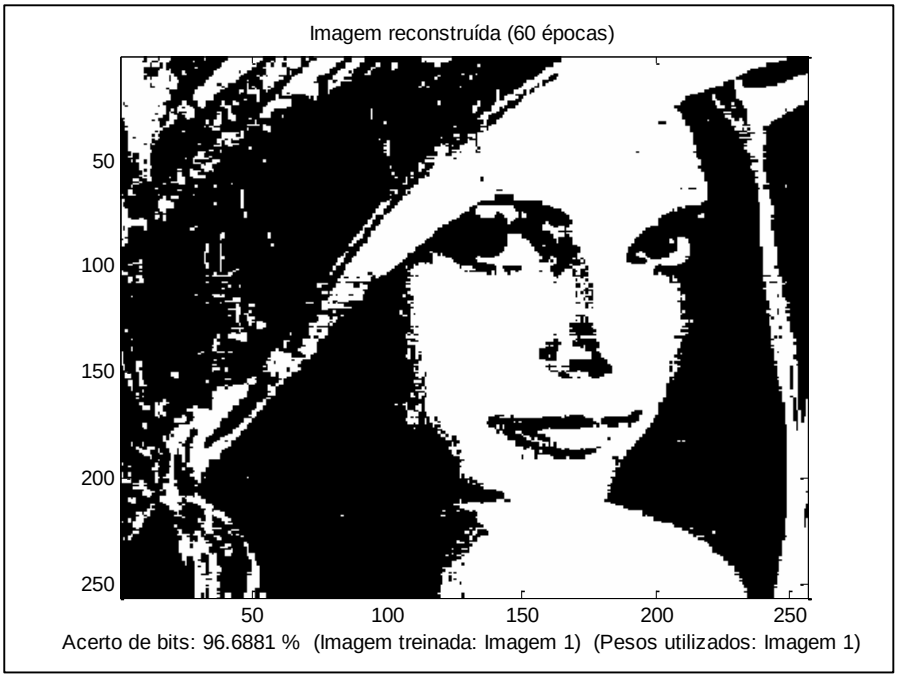

Figura 5.9 - Imagem da Lenna reconstruída com 60 épocas a partir dos pesos da Imagem 1.

Na figura 5.10, o gráfico demonstra a evolução do percentual de acertos de bits da imagem reconstruída conforme aumentou-se o número de épocas.

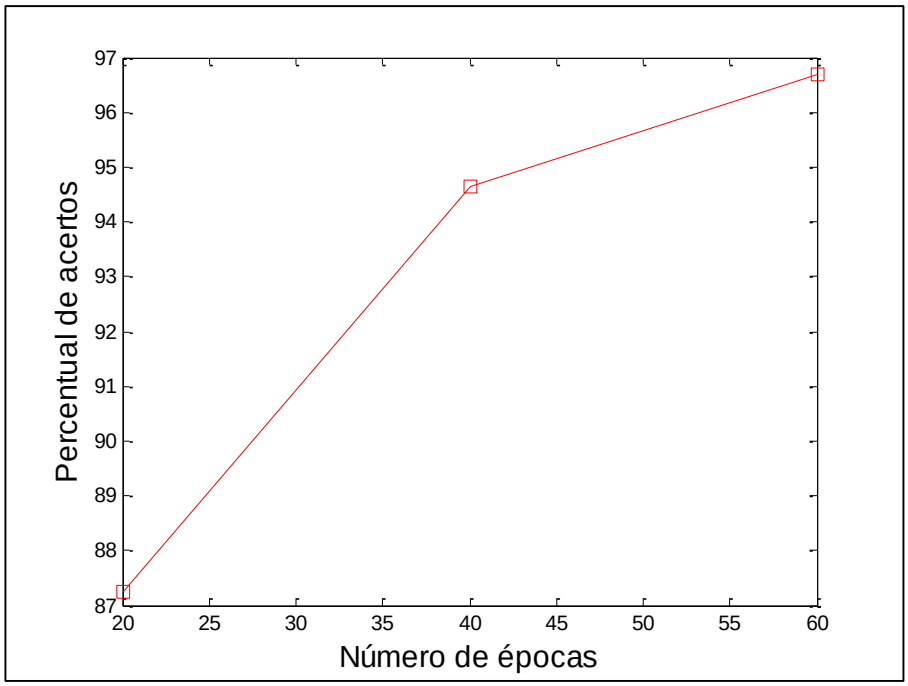

Figura 5.10 - Gráfico do percentual de acertos de bits da imagem da Lenna reconstruída utilizando pesos da Imagem 1 com diferentes números de épocas.

Nestes primeiros treinamentos, foi possível notar a capacidade da rede em reconstruir imagens. Para testar a capacidade da rede em classificar padrões, foi realizado testes de reconstrução de padrão com pesos treinados de outras imagens. Por exemplo, na figura 5.11 utilizou-se a Imagem 1 no algoritmo porém os pesos treinados foram da Imagem 2. Com 20 épocas de treinamento, o percentual de acerto de bits foi 74,94%. Na figura 5.12, com 40 épocas de treinamentos, o percentual de acertos caiu para 73,76%. Por fim na figura 5.13, com 60 épocas de treinamento, o percentual de acertos foi de 74,08%. Na figura 5.14, é apresentado um gráfico comparando o percentual de acerto de bits para os diferentes números de épocas.

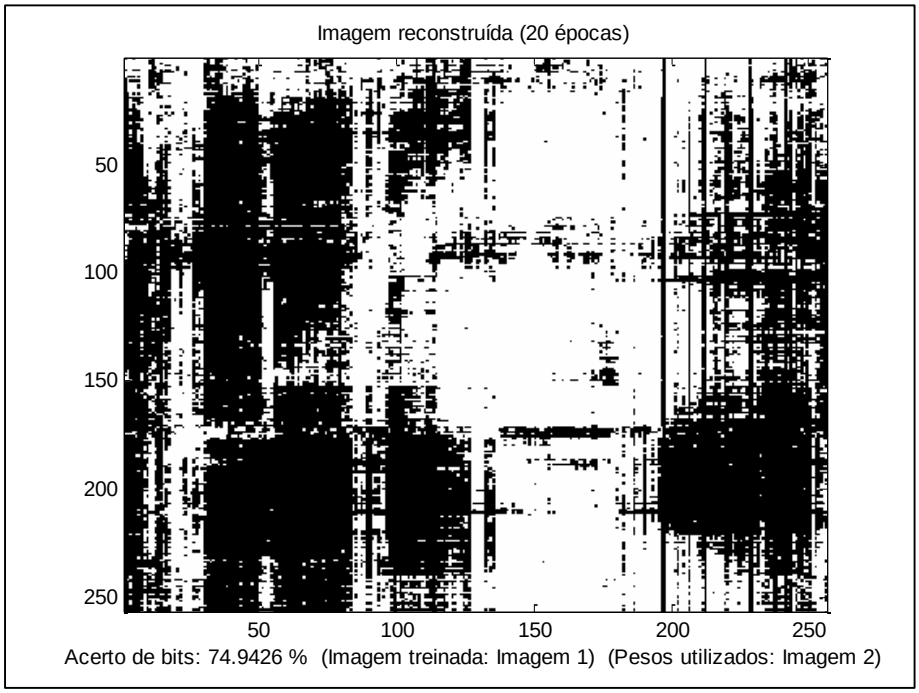

Figura 5.11 - Imagem da Lenna reconstruída com 20 épocas a partir dos pesos da Imagem 2.

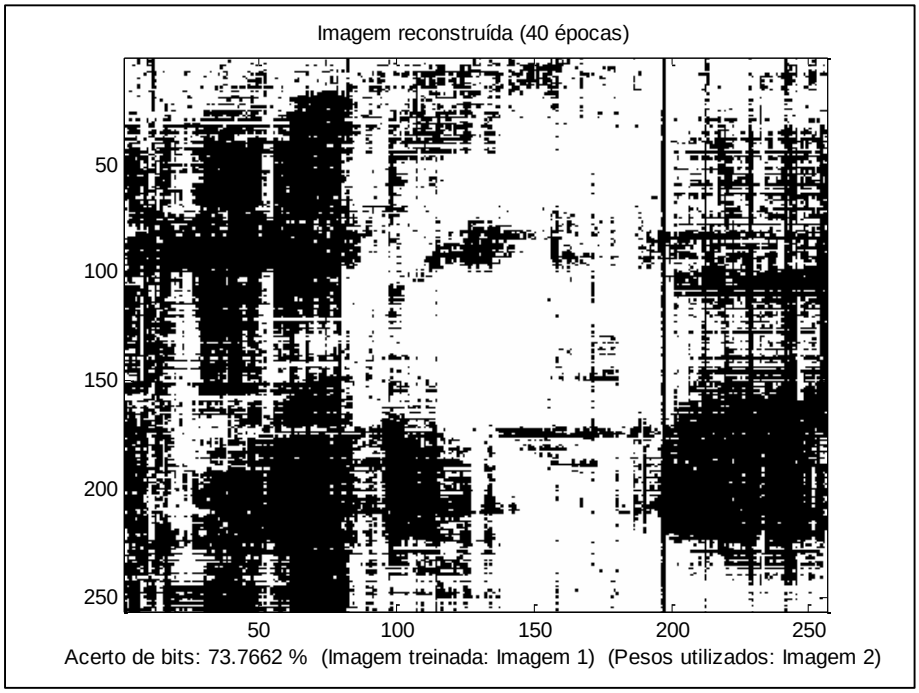

Figura 5.12 - Imagem da Lenna reconstruída com 40 épocas a partir dos pesos da Imagem 2.

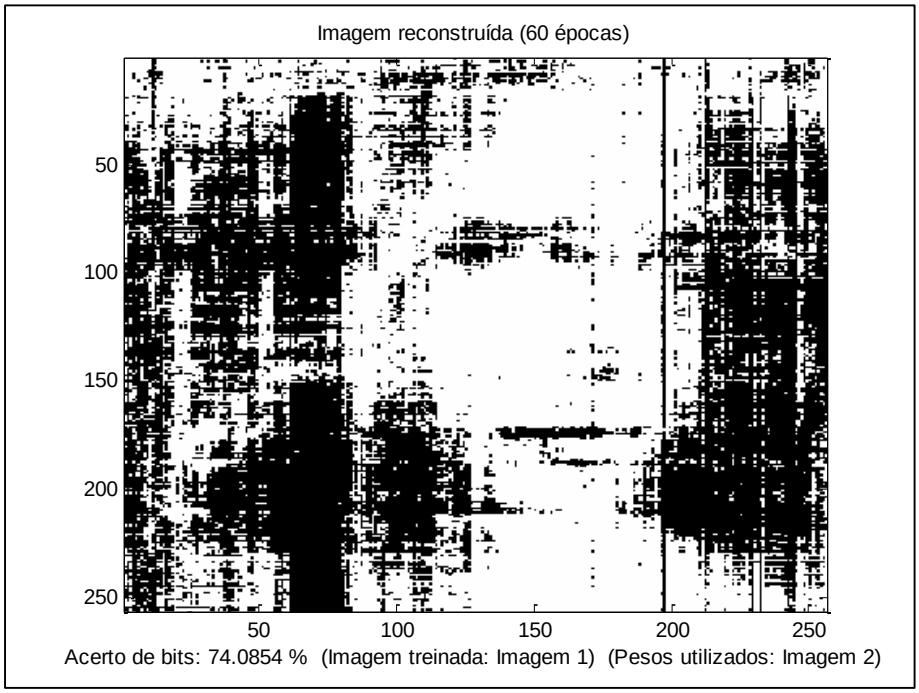

Figura 5.13 - Imagem da Lenna reconstruída com 60 épocas a partir dos pesos da Imagem 2.

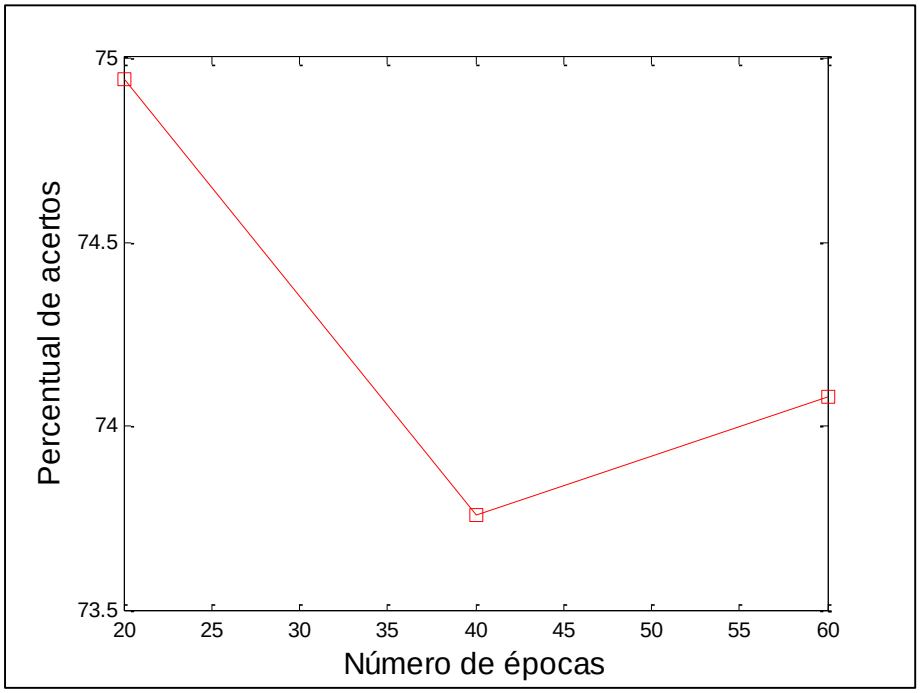

Figura 5.14 - Gráfico do percentual de acertos de bits da imagem da Lenna reconstruída utilizando pesos da Imagem 2 com diferentes números de épocas.

Novos testes foram realizados, desta vez utilizou-se a Imagem 1 no algoritmo RBM porém os pesos treinados foram da Imagem 3. Na figura 5.15, com 40 épocas de treinamento o percentual de acertos foi de 73,88 %. Já na figura 5.16, com 60 épocas o percentual de acertos foi de 72,17%.

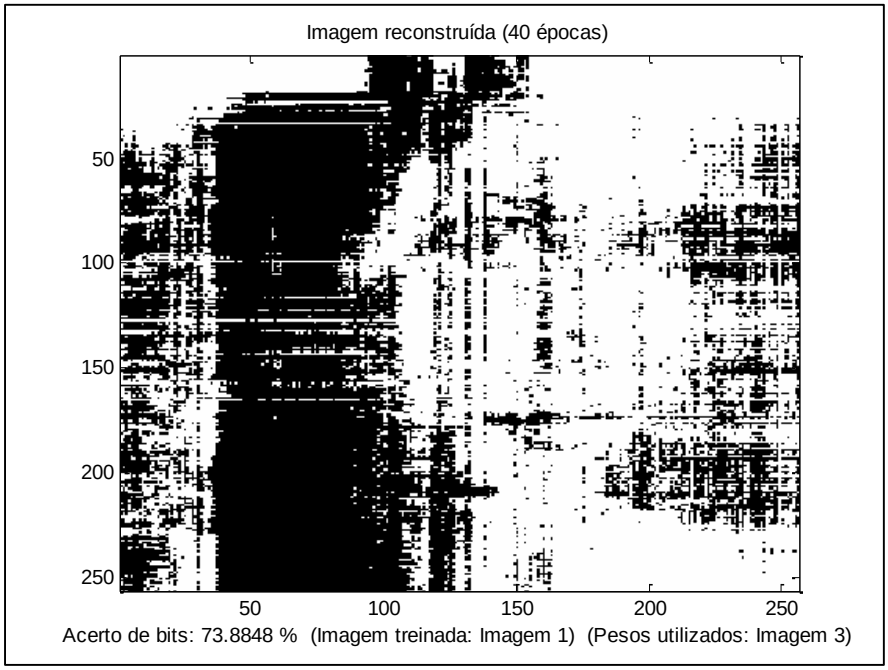

Figura 5.15 - Imagem da Lenna reconstruída com 40 épocas a partir dos pesos da Imagem 3.

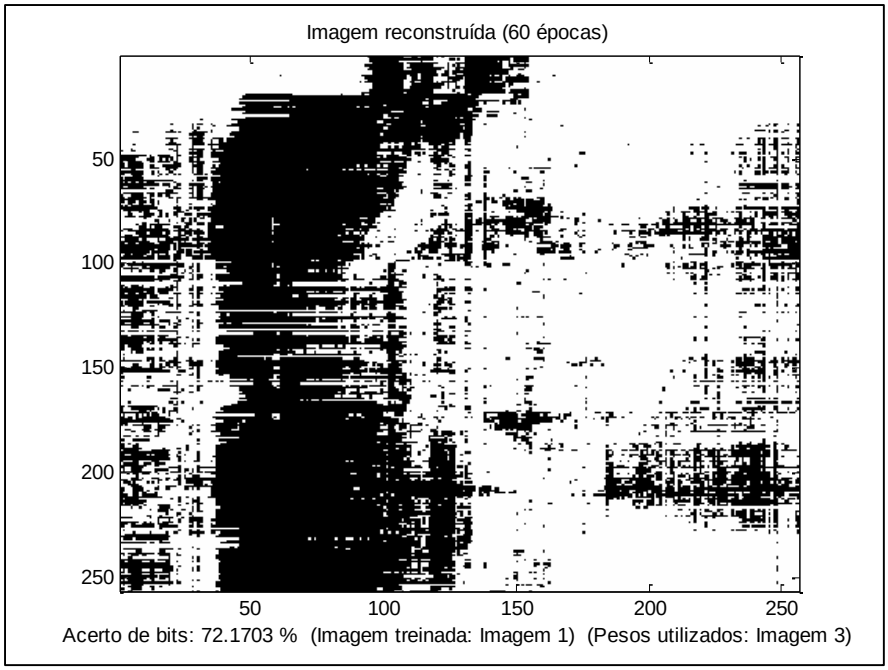

Figura 5.16 - Imagem da Lenna reconstruída com 60 épocas a partir dos pesos da Imagem 3.

Ao final destes treinamentos realizados foi possível comparar o percentual de acertos das imagens reconstruídas quando se utilizou os pesos treinados da própria imagem e utilizando os pesos treinados de outras imagens conforme a figura 5.17 a seguir. Através deste percentual de acertos é possível classificar a imagem através do algoritmo RBM; pois enquanto os pesos da imagem correta tiveram um percentual de acertos superior a 90%, os pesos da imagem incorreta tiveram um percentual de acertos em torno de 70%.

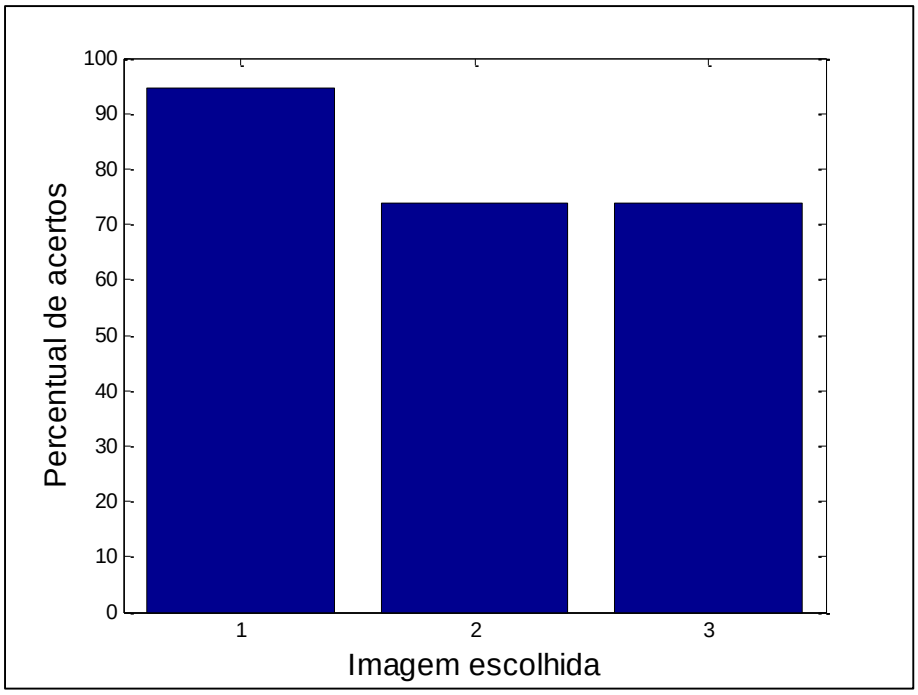

Figura 5.17 - Gráfico comparativo do percentual de acertos de bits da Imagem 1 reconstruída utilizando pesos das 3 imagens.

Em seguida foi realizado novos testes utilizando a imagem da Ponte chamada de Imagem 2. Primeiramente a imagem foi binarizada conforme figura 5.18 e foi inserido um ruído aleatório exibido na figura 5.19.

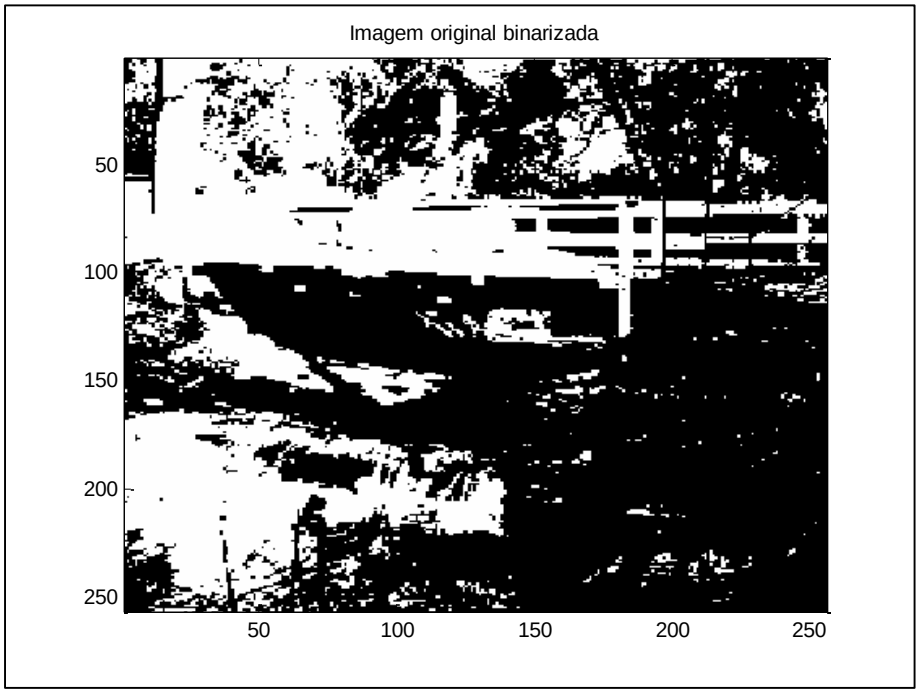

Figura 5.18 - Imagem da Ponte binarizada.

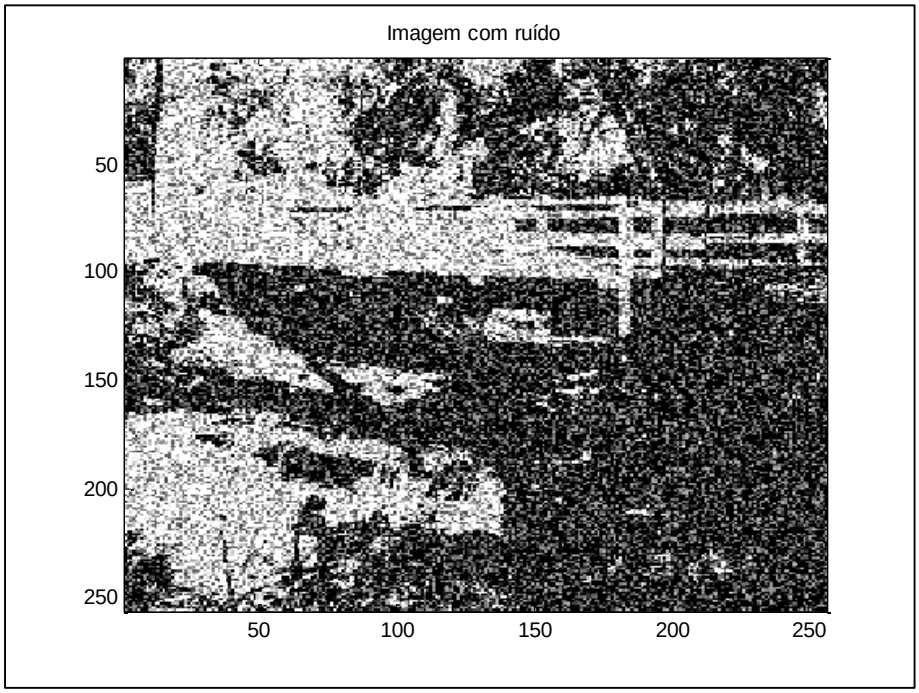

Figura 5.19 - Imagem da Ponte com inserção de ruído aleatório.

Após o treinamento da imagem através do algoritmo RBM, a imagem foi reconstruída com os pesos treinados em 20, 40 e 60 épocas. Os resultados da reconstrução da imagem são apresentados respectivamente através das imagens 5.20, 5.21 e 5.22 com os percentuais de

acerto de 95,84%, 97,67% e 97,92%. A figura 5.23 apresenta esta evolução do percentual de acerto de bits conforme o número de épocas.

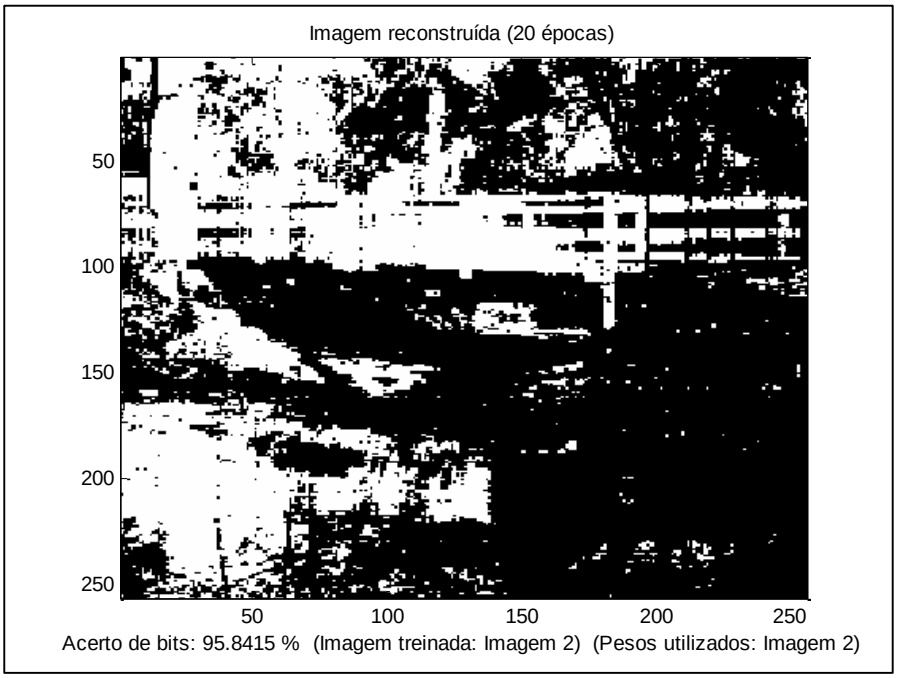

Figura 5.20 - Imagem da Ponte reconstruída com 20 épocas a partir dos pesos da Imagem 2.

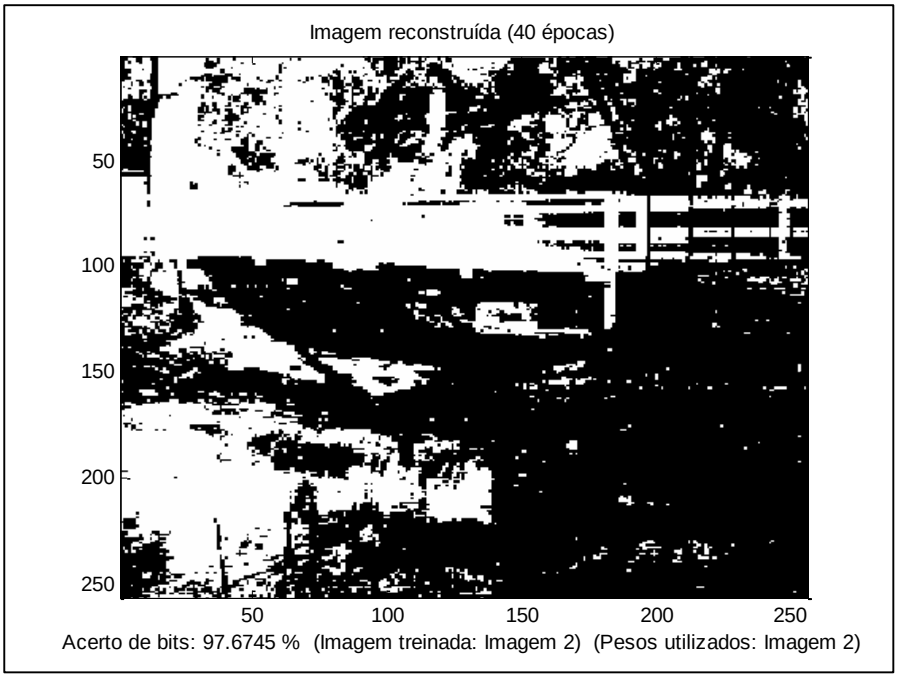

Figura 5.21 - Imagem da Ponte reconstruída com 40 épocas a partir dos pesos da Imagem 2.

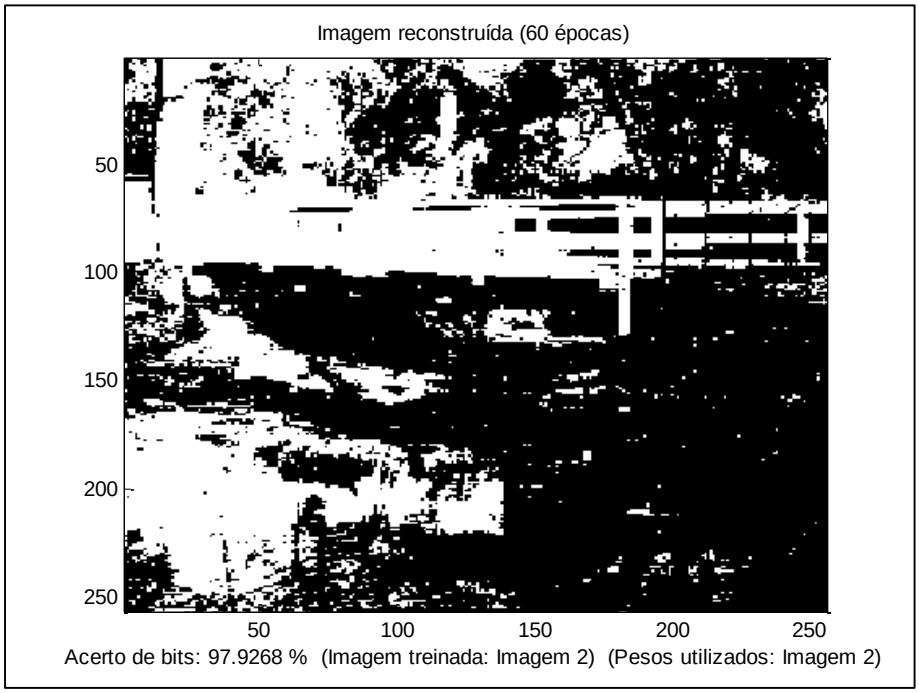

Figura 5.22 - Imagem da Ponte reconstruída com 60 épocas a partir dos pesos da Imagem 2.

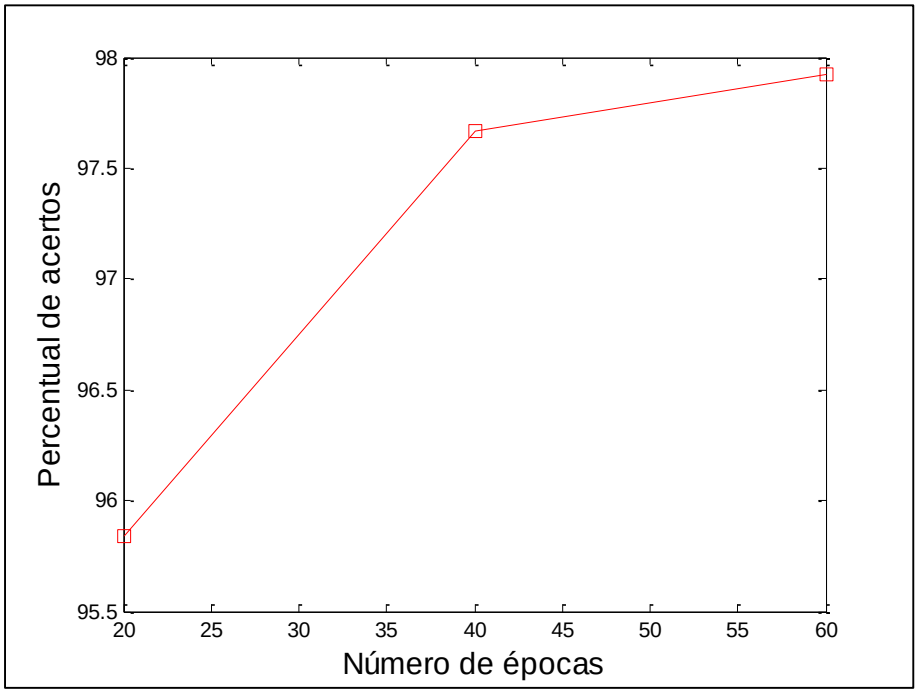

Figura 5.23 - Gráfico do percentual de acertos de bits da imagem da Ponte reconstruída utilizando pesos da Imagem 2 com diferentes números de épocas.

Com esta imagem da ponte, também foi testada a capacidade da rede em classificar padrões. Para isto, fui utilizada a imagem 2 no algoritmo RBM mas os pesos treinados foram das imagens 1 e 3, conforme figuras as 5.24 e 5.25, com os percentuais de acertos de 72,58% e 65,32%, respectivamente.

Através destas execuções do algoritmo, foi possível perceber que quando se utilizou os pesos treinados da Imagem 2 para a reconstrução desta mesma Imagem 2, houve um acerto superior a 90% e quando se utilizou pesos de outras imagens, houve um acerto em torno de 70%. O gráfico da figura 5.26 apresenta este comparativo do percentual de acerto de acordo com os pesos utilizados para reconstrução.

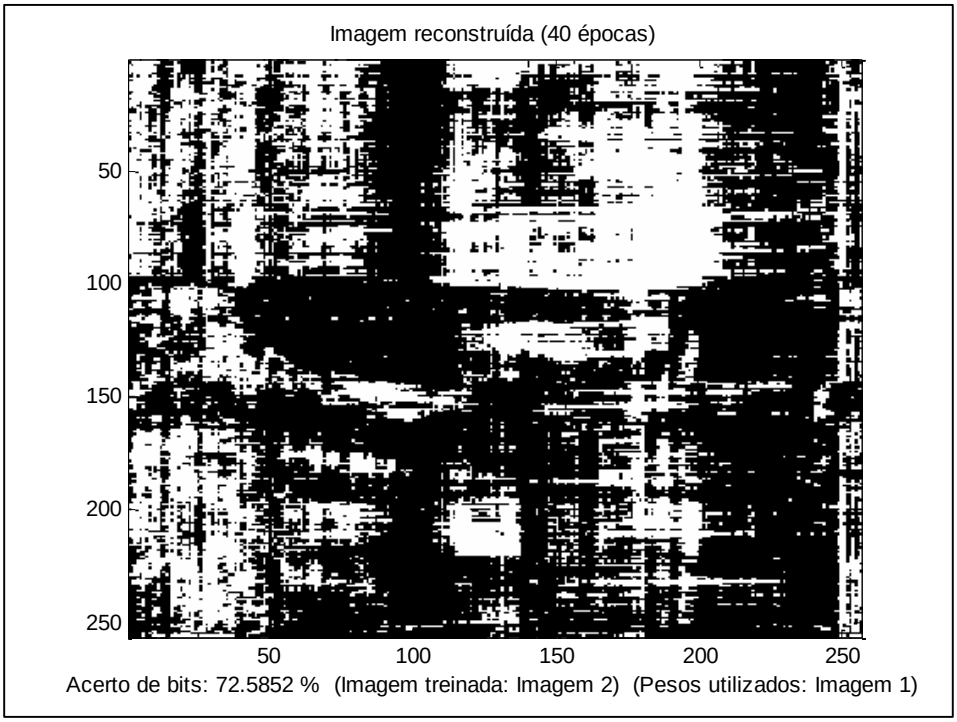

Figura 5.24 - Imagem da Ponte reconstruída com 40 épocas a partir dos pesos da Imagem 1.

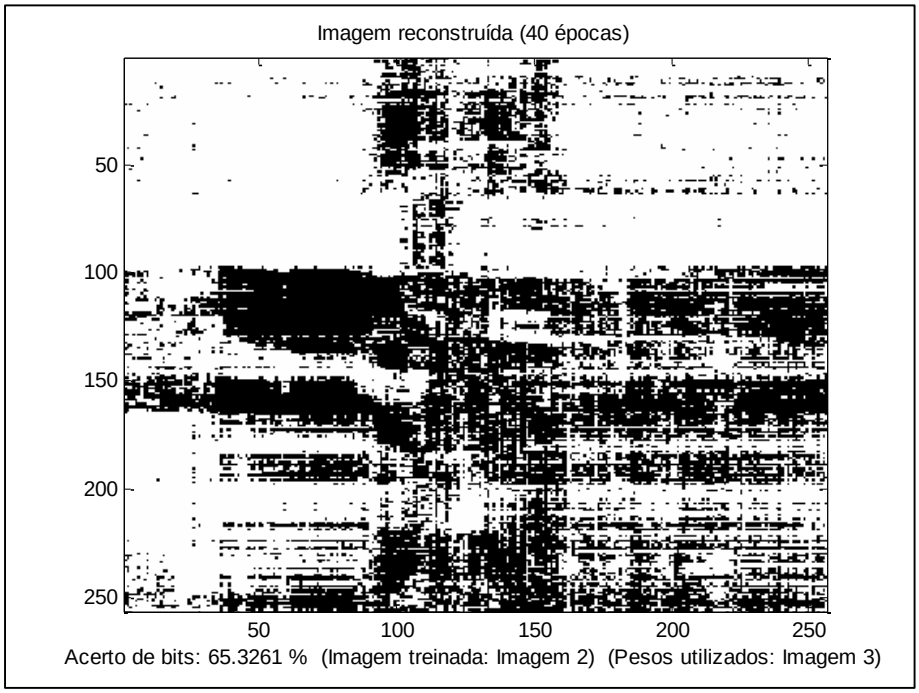

Figura 5.25 - Imagem da Ponte reconstruída com 40 épocas a partir dos pesos da Imagem 3.

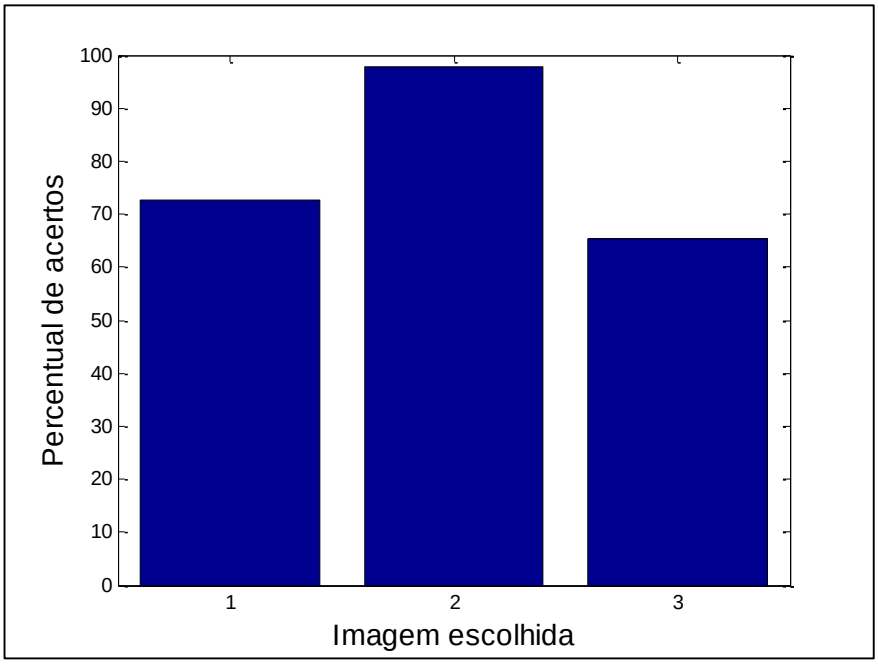

Figura 5.26 - Gráfico comparativo do percentual de acertos de bits da Imagem 2 reconstruída utilizando pesos das 3 Imagens.

Por fim, os mesmos testes foram executados com a imagem do Câmera, Imagem 3. A figura 5.27 mostra a imagem binarizada e a figura 5.28 exibe a imagem com inserção de ruído.

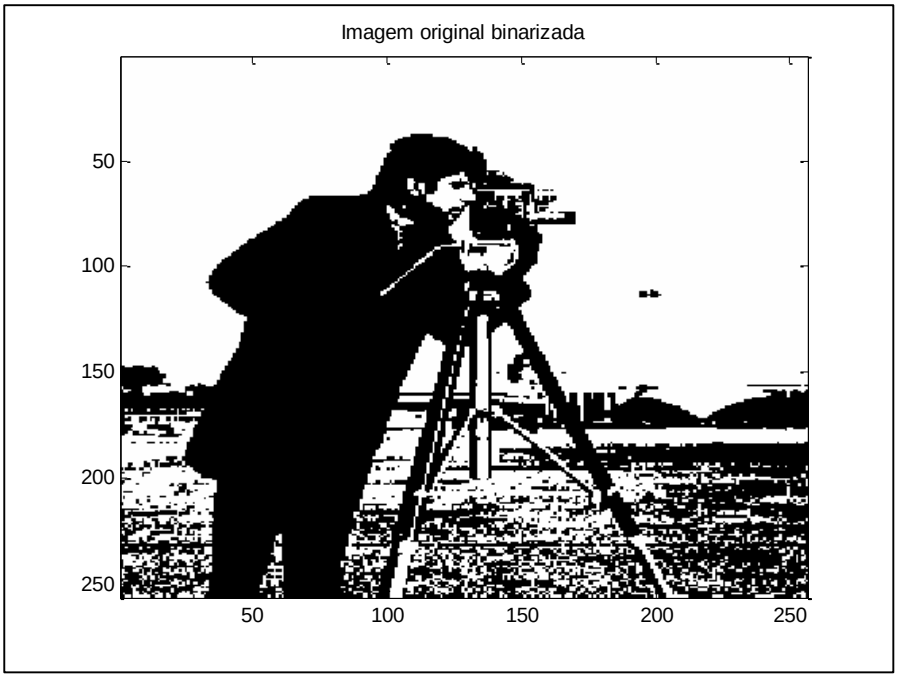

Figura 5.27 - Imagem do Câmera binarizada.

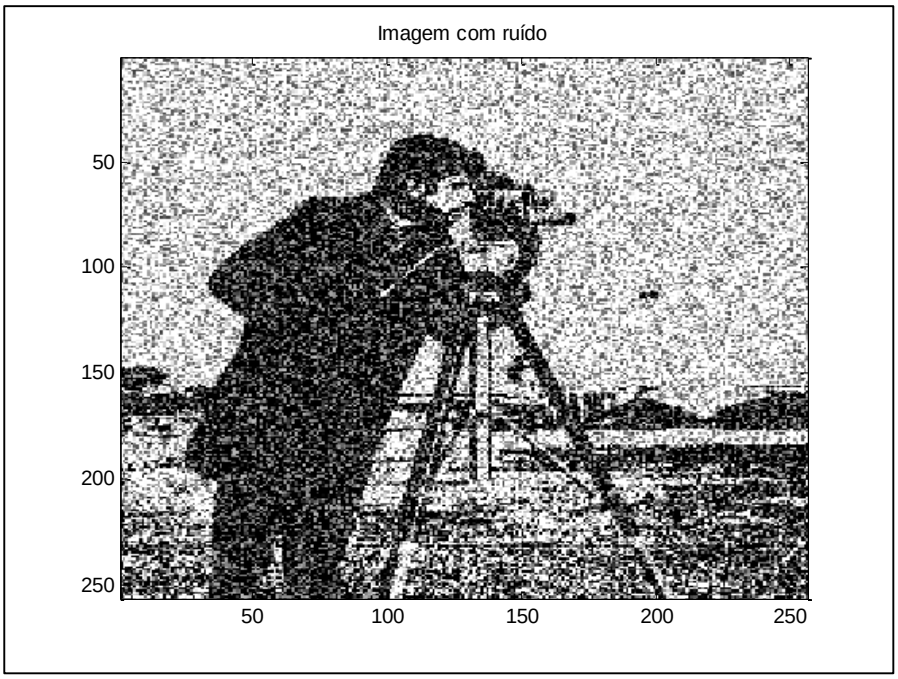

Figura 5.28 - Imagem do Câmera com inserção de ruído aleatório.

Nas figuras 5.29, 5.30 e 5.31 são exibidas as imagens reconstruídas após a aplicação do algoritmo RBM em 20, 40 e 60 épocas, respectivamente. As taxas de acerto de bits foram de 95,69%, 97,68% e 98,04%, conforme nota-se no gráfico comparativo da figura 5.32.

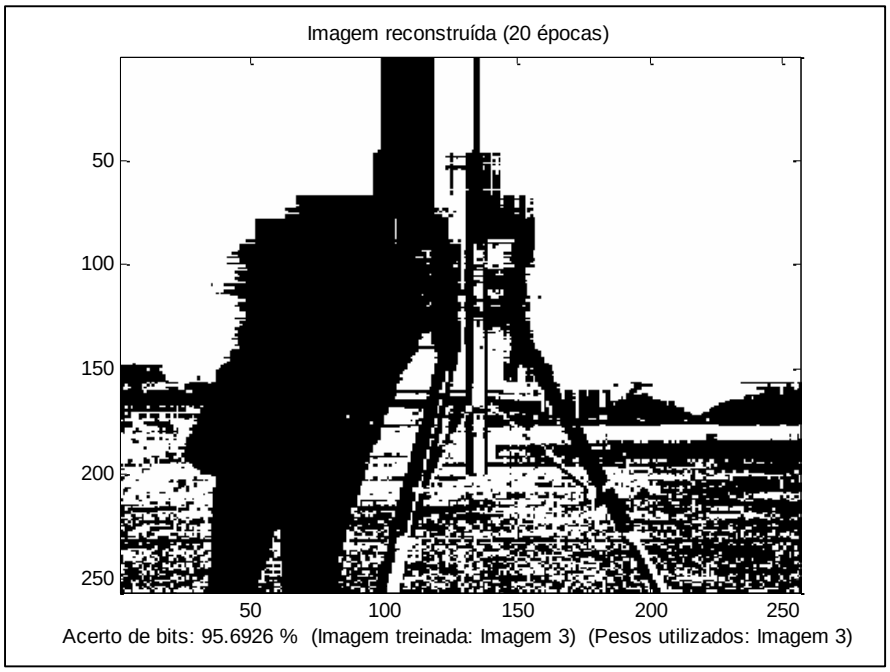

Figura 5.29 - Imagem do Câmera reconstruída com 20 épocas a partir dos pesos da Imagem 3.

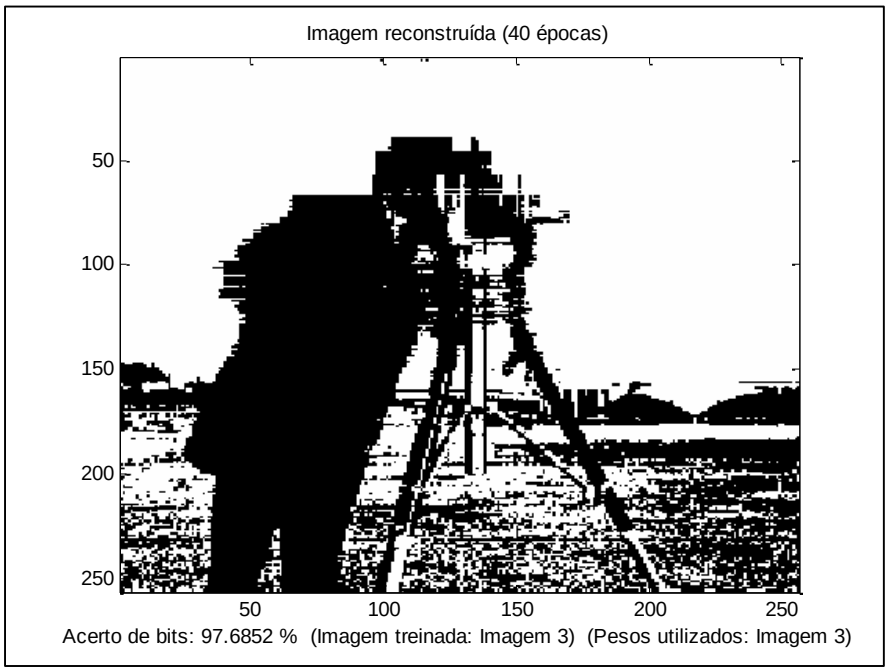

Figura 5.30 - Imagem do Câmera reconstruída com 40 épocas a partir dos pesos da Imagem 3.

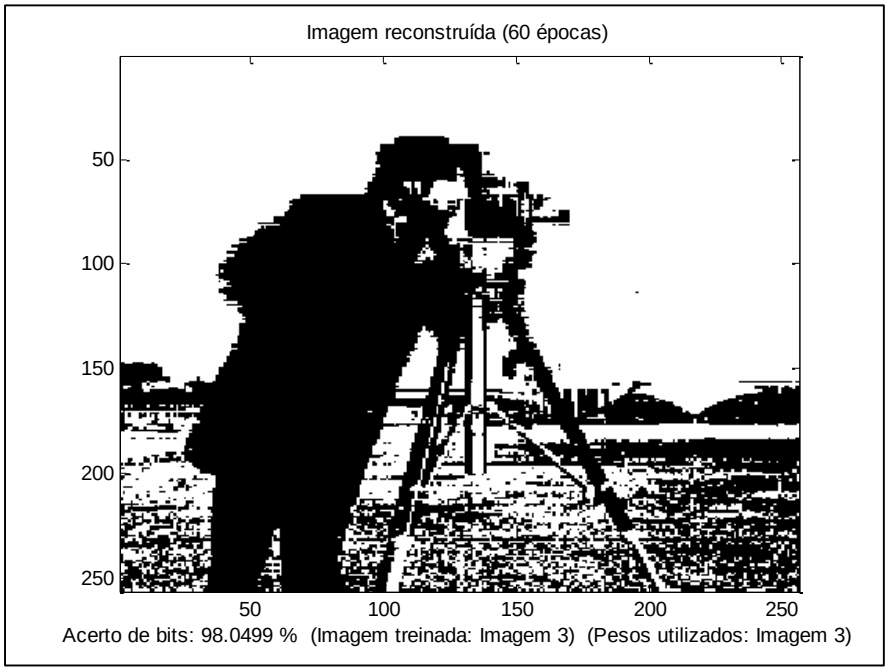

Figura 5.31 - Imagem do Câmera reconstruída com 60 épocas a partir dos pesos da Imagem 3.

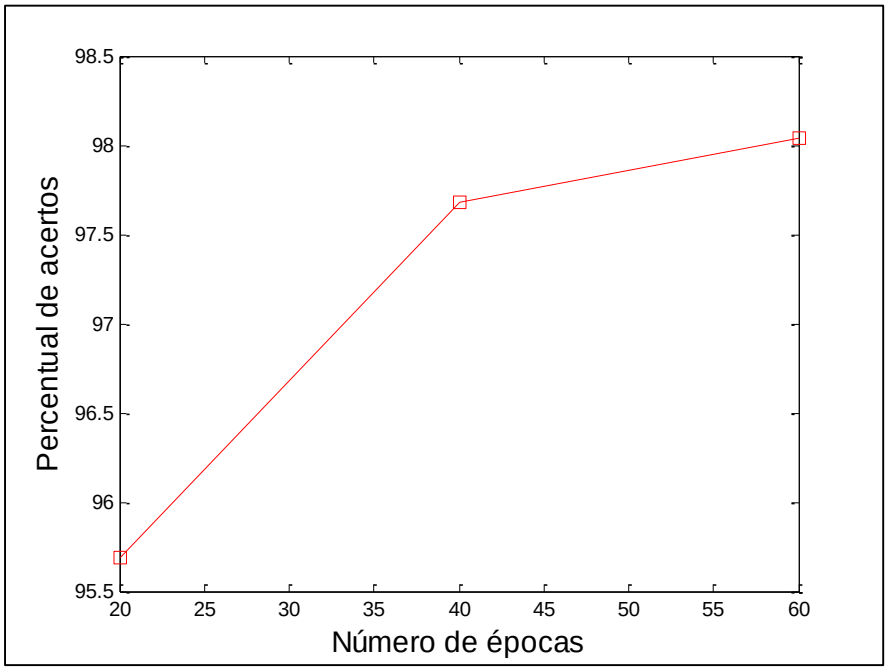

Figura 5.32 - Gráfico do percentual de acertos de bits da imagem do Câmera reconstruída utilizando pesos da Imagem 3 com diferentes números de épocas.

A fim de se testar a capacidade de classificação das imagens, também foram feitos testes de reconstrução com pesos treinados de outras imagens. Na figura 5.33, a imagem foi reconstruída com os pesos treinados da Imagem 1, resultando um percentual de acertos de bits de 73,15%. Na figura 5.34, utilizou-se os pesos treinados de Imagem 2, tendo um percentual de 76,33%.

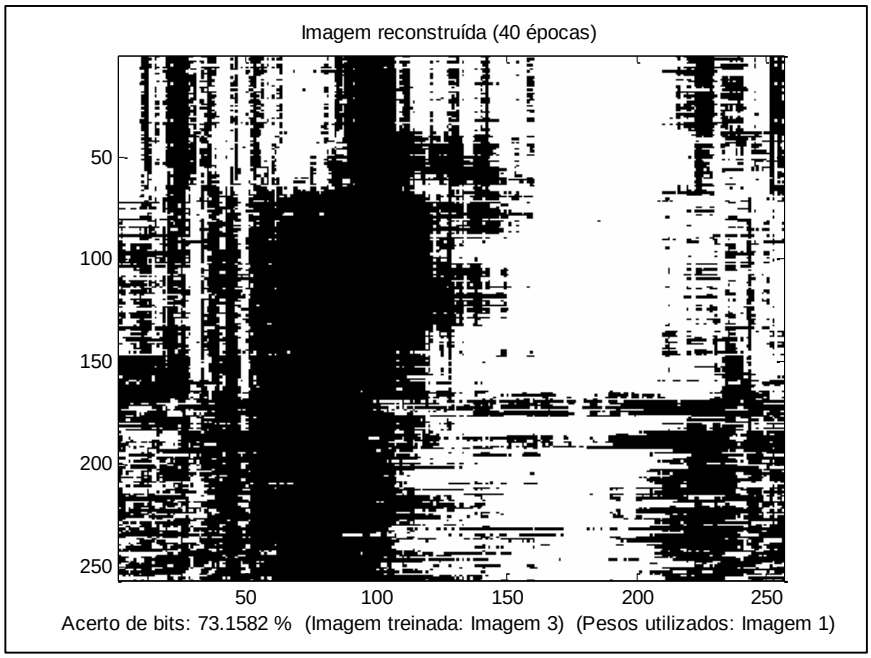

Figura 5.33 - Imagem do Câmera reconstruída com 40 épocas a partir dos pesos da Imagem 1.

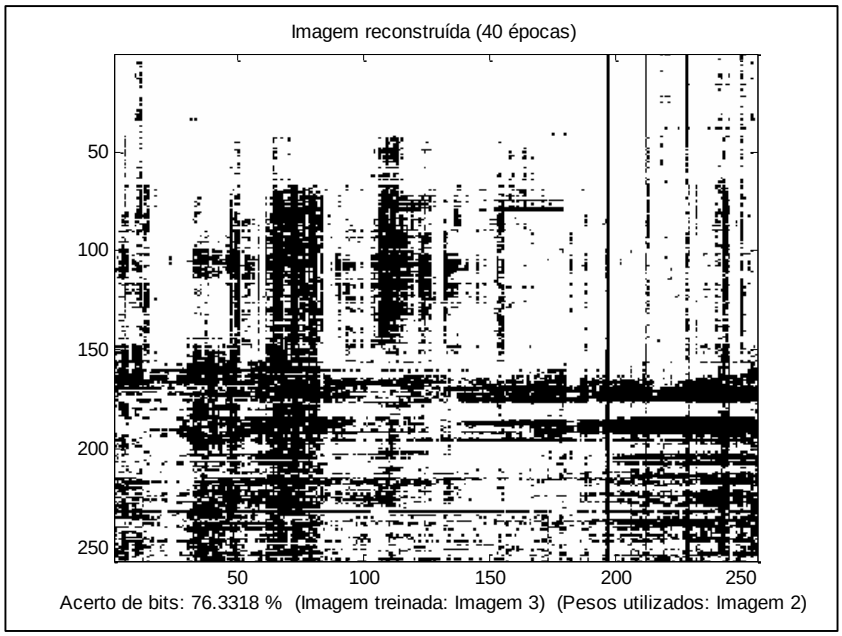

Figura 5.34 - Imagem do Câmera reconstruída com 40 épocas a partir dos pesos da Imagem 2.

Novamente, comparando os resultados é possível classificar a imagem reconstruída através dos resultados que apresentaram o maior percentual de acerto de bits. O gráfico da figura 5.35 apresenta o comparativo do percentual de acerto de bits utilizando pesos treinados de diferentes imagens.

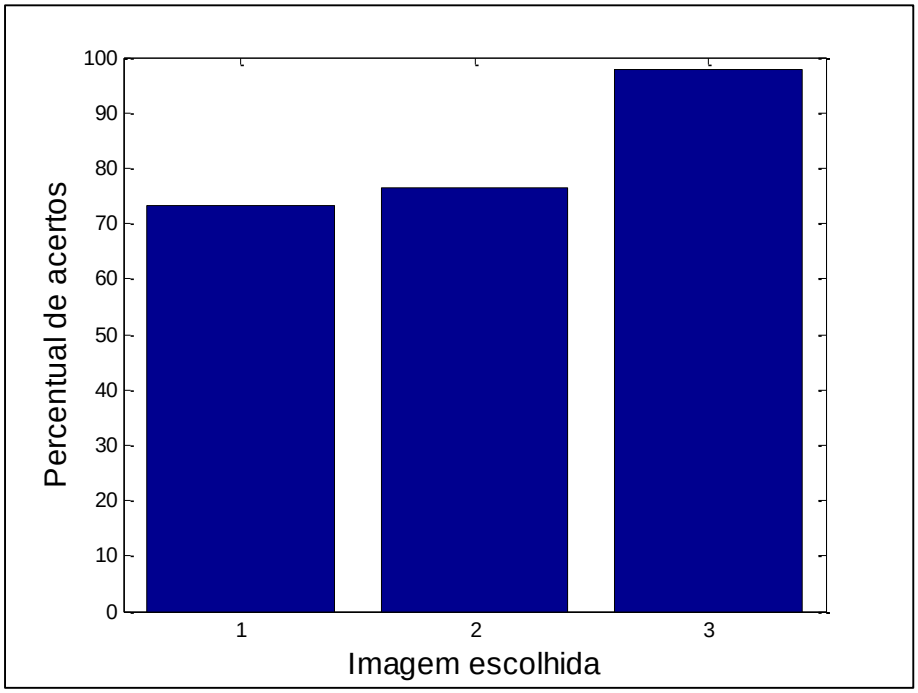

Figura 5.35 - Gráfico comparativo do percentual de acertos de bits da Imagem 3 reconstruída utilizando pesos das 3 imagens.

5.3. CONSIDERAÇÕES FINAIS DESTE CAPÍTULO

Neste capítulo foi utilizado o algoritmo RBM em reconstrução e classificação de imagens. Foram utilizadas 3 imagens com ruído e testou-se a capacidade de reconstrução de imagens com os pesos treinados da própria imagem e com os pesos treinados das outras imagens.

Desta forma, foi possível constatar que o algoritmo RBM foi capaz de reconstruir imagens ruidosas com um grande percentual de acertos e uma boa relação sinal-ruído. Através dos percentuais de acerto de *bits* é possível classificar a imagem reconstruída como foi observado em várias simulações neste capítulo.

CAPÍTULO 6

6. CONCLUSÕES E TRABALHOS FUTUROS

6.1. DESENVOLVIMENTO DO TRABALHO

O Capítulo 1 deste trabalho apresentou uma introdução da pesquisa que foi desenvolvida. Neste capítulo foi descrita a importância da reconstrução e classificação de padrões principalmente de imagens.

O Capítulo 2 apresentou os principais fundamentos teóricos das Redes Neurais Artificiais. Foram descritos o histórico e o funcionamento de cada uma das principais RNAs, como: P*erceptrons, Adaline, Backpropagation* e o modelo de *Hopfield*. As principais estruturas e processos utilizados para treinamento e aprendizagem de padrões foram descritos neste capítulo.

O Capítulo 3 apresentou o algoritmo Máquina de *Boltzmann* para reconhecimento e classificação de padrões. Primeiramente foi descrito o algoritmo Máquina de *Boltzmann* Geral com a utilização da técnica de recozimento simulado e, em seguida, foi descrito o algoritmo de aprendizagem Máquina de *Boltzmann* Restrita.

O Capítulo 4 comparou simulações de reconstrução e classificação de padrões utilizando os diferentes algoritmos descritos em capítulos anteriores, como: RNA *Backpropagation*, Máquina de *Boltzmann* Geral e Máquina de *Boltzmann* Restrita. Os padrões utilizados como entrada para as simulações foram os padrões de um display de 7 segmentos e sequências binárias.

Finalmente, o Capítulo 5 apresentou simulações utilizando o algoritmo RBM para reconstrução de imagens binárias e para classificação de padrões. Para esta comparação foram utilizadas 3 diferentes imagens binárias e foram comparados os percentuais de acertos de *bits* na recuperação das imagens. Através destes percentuais de acerto foi possível classificar corretamente as imagens reconstruídas pelo algoritmo RBM.

6.2. CONCLUSÕES DESTA PESQUISA

Nesta pesquisa realizada foi possível comparar os algoritmos RNAs que são utilizados com maior frequência para reconhecimento de padrões com os algoritmos Máquina de

Boltzmann Geral e Máquina de *Boltzmann* Restrita. As simulações computacionais realizadas testaram a eficácia destes algoritmos que possuem características estocásticas e probabilísticas e certificaram a capacidade destes processos em reconstruir padrões treinados por estas redes.

Também através destas simulações foi possível notar que o algoritmo Máquina de *Boltzmann* Geral apresentou melhores resultados em classificar os padrões de entrada simulados, levando-se em consideração o tempo de execução do código e o percentual de acertos. Através dos gráficos e tabelas apresentadas ao longo deste trabalho, é nitidamente perceptível que o algoritmo Máquina de *Boltzmann* Geral teve um melhor percentual de acertos de *bits* e um melhor tempo de execução do código comparado ao algoritmo RNA *Backpropagation*.

Por fim, foi treinada na rede Máquina de *Boltzmann* Restrita imagens binárias com e sem ruído, e esta rede conseguiu reconstruir estas imagens binárias com um grande percentual de acerto de *bits* quando comparado com as imagens originais. Assim, este algoritmo Máquina de *Boltzmann* Restrita mostrou-se capaz de reconstruir e classificar imagens. Desta forma, ele pode ser amplamente utilizado em outras aplicações práticas que requerem o reconhecimento de padrões como de faces, íris do olho, escritas e outros.

6.3. TRABALHOS FUTUROS

Os resultados obtidos neste trabalho podem ser aplicados em problemas mais complexos envolvendo classificação de imagens. O trabalho desenvolvido utilizou como entrada uma quantidade reduzida de dados pois utilizou imagens binárias de baixa resolução, porém pode ser aplicado em problemas mais complexos.

Além disso, o algoritmo Máquina de *Boltzmann* auxilia na diminuição da carga computacional para situações que exigem muitas variáveis de entrada para o reconhecimento. Estes algoritmos poderiam ser aplicados em outros trabalhos tais como: reconhecimento de faces, preenchimento de padrões defeituosos, reconhecimento de íris, identificação de escrita e demais padrões. Nota-se também neste trabalho, que as reconstruções de imagens foram realizadas com imagens ruidosas; estas reconstruções poderiam ser simuladas com imagens sombreadas, imagens com ausência de padrões e outras condições que necessitariam de reconstrução e classificação de imagens.

Estes algoritmos estudados são de pesquisas recentes e por isso ainda possuem muito campo para exploração podendo ser ampliados em trabalhos futuros e aplicados em muitas das situações descritas anteriormente.

REFERÊNCIAS

[1] L. Fausett, Fundamentals of Neural Networks: Architectures, algorithms, and Applications, Florida: Prentice Hall, 1994.

[2] J. M. N. Silva, "Redes Neurais Artificiais: Rede Hopfield e Redes Estocásticas," Nitéroi, 2003.

[3] A. d. P. Braga, A. P. d. L. F. d. Carvalho e T. B. Ludermir, Redes Neurais Artificiais: Teoria e Aplicações, Rio de Janeiro: LTC - LIVROS TÉCNICOS E CIENTÍFICOS EDITORA S.A., 2000.

[4] M. Minsky e S. A. Papert, Perceptrons: An Introduction to Computational Geometry, Cambridge : MIT PRESS, 1987.

[5] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," em *Proceding of the National Academy of Sciences*, 1982.

[6] J. J. Hopfield, "Neurons with Graded Response Have Collective Computational Properties like Those of Two-state Neurons," em *Proceedings of the National Academy of Sciences*, 1984.

[7] D. H. Ackley, G. E. Hinton e T. J. Sejnowski, "A Learning Algorithm for Boltzmann Machines*," *Cognitive Science,* p. 147–169, Janeiro 1985.

[8] D. E. G. E. H. &. R. J. W. RuMELHART, "Learning Representations by Back-Propagating Error," *Nature,* vol. 323, pp. 533-536, Outubro 1986.

[9] W. P. Warren S. McCulloch, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics,* pp. 18-28, Dezembro 1943.

[10] T. W. Rauber, "Redes Neurais Artificiais," Vitória.

[11] D. D. d. Silva, "A Utilização de Redes Neurais Artificiais na Estimação da Cobertura do Sinal de Televisão Digital," Goiânia, 2009.

[12] G. Cybenko, "Approximation by superpositions of a sigmoid function," *Mathematics of Control, Signals and Systems,* pp. 2:303-314, Dezembro 1989.

[13] G. Cybenko, “Continuos valued neural networks with two hidden layers ar sufficient,” 1988.

[14] G. E. Hinton e T. J. Sejnowski, “Optimal Perceptual Inference,” em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Washington, 1983.

[15] S. P. Parker, “GPU implementation of a deep learning network for image recognition tasks,” Iowa, 2012.

[16] S. Sivanandam e S. N. Deepa, Introduction to Neural Networks Using Matlab 6.0, India: Tata McGraw-Hill Education, 2006, p. 656.

[17] R. O. Duda, P. E. Hart e D. G. Stork, Pattern Classification, Second Edition ed., New York: John Wiley & Sons, 2001.

[18] R. Salakhutdinov e G. E. Hinton, “Deep boltzmann machines,” em *Proceedings of the international conference on artificial intelligence and statistics*, Cambridge, 2009.

[19] A. Fischer e C. Igel, “Training Restricted Boltzmann Machines: An Introduction,” *Pattern Recognition*, vol. 47, pp. 25-39, January 2014.

[20] J. Melchior, “Learning Natural Image Statistics with Gaussian-Binary Restricted Boltzmann Machines,” Bochum, 2012.

[21] X. Wang, V. Ly, R. Guo e C. Kambhamettu., “2D-3D Face Recognition via Restricted Boltzmann Machines”.

[22] C. S. Miranda, “Proposta de Modelo Semiótico Baseado em Máquinas de Boltzmann Restritas,” em *4o. Seminário Interno de Semiótica e Sistemas Inteligentes - SISSI 2012*, Campinas, 2012.

[23] R. Salakhutdinov e G. Hinton, “Deep Boltzmann Machines”.