Universidade Federal do Ceará
Departamento de Engenharia Elétrica
Curso de Graduação em Engenharia Elétrica

Alba María De Castro Carrascosa

Modelagem do Robô Móvel SCITOS G5 Usando a Rede Neural ELM: Um Estudo de Caso para as Velocidades de Rotação e Translação

Fortaleza – Ceará
Junho 2011

Autor:
Alba María De Castro Carrascosa

Orientador:
Prof. Dr. Guilherme de Alencar Barreto

Modelagem do Robô Móvel SCITOS G5 Usando a Rede Neural ELM: Um Estudo de Caso para as Velocidades de Rotação e Translação

Trabalho de Conclusão de Curso apresentada à Coordenação do Curso de Graduação em Engenharia Elétrica da Universidade Federal do Ceará como parte dos requisitos para obtenção do grau de **Engenharia de Teleinformática**.

Fortaleza – Ceará
Junho 2011

Alba María De Castro Carrascosa

Modelagem do Robô Móvel SCITOS G5 Usando a Rede Neural ELM: Um Estudo de Caso para as Velocidades de Rotação e Translação

Este trabalho foi julgado adequado para a obtenção do diploma de Engenheiro do Curso de Graduação em Engenharia de Teleinformática da Universidade Federal do Ceará.

Alba María De Castro Carrascosa

Banca Examinadora:

Prof. Dr. Guilherme de Alencar Barreto
Orientador

Prof. MSc. Carlos Pimentel de Souza

Prof. Dr. George André Pereira Thé

Fortaleza, 3 de junho de 2011

Resumo

Esta monografia reporta os resultados gerados pela aplicação da rede ELM (Extreme Learning Machine) no aprendizado das velocidades de translação e rotação do robô móvel SCITOS G5 em função das leituras dos sensores de ultrassom. As leituras dos 24 sensores ultrassom (entradas) e as respectivas velocidades de rotação e translação (saídas) para um certo instante de amostragem foram coletadas durante a execução de uma tarefa de navegação do tipo Wall-following (seguir paredes). Um total de 5456 pontos foram coletados, correspondendo a aproximadamente 4 voltas em sentido horário em torno da sala onde o experimento foi realizado. De posse dos dados, diferentes arquiteturas da rede ELM(p,q,m) foram treinadas usando os primeiros 4000 pontos (aproximadamente 73% do total), em que $p \in \{2, 4, 24\}$ define o número de unidades de entrada, $q \in \{5, 20, 50, 100, 200\}$ denota o número de neurônios escondidos e $m \in \{1, 2\}$ indica o número de neurônios de saída. O restante dos pontos foi usado para teste das arquiteturas treinadas. Nos testes realizados, os neurônios ocultos usaram a função de ativação sigmóide logística e os neurônios de saída são lineares. Foram realizados vários testes a fim de avaliar o desempenho da rede ELM no aprendizado das velocidades de rotação e translação para diferentes dimensões do vetor de entrada (p) e para diferentes quantidades de neurônios ocultos. De um modo geral, a rede ELM foi capaz de "aprender"(modelar) a relação entrada-saída para a velocidade de translação, com o melhor resultado obtido para $p = 2$ e $q = 200$. Já no caso da velocidade de rotação, a rede ELM teve um desempenho muito abaixo do esperado, não sendo capaz de aprender a relação entrada-saída correspondente.

Palavras-chaves: ELM, robô SCITOS G5, sensores ultrasom, redes reurais, Wall-following.

Abstract

This monograph reports the results obtained from the application of the ELM (Extreme Learning Machine) neural network to the problem of learning the rotational and translational velocities for the mobile robot SCITOS G5, as a function of the ultrasound sensor readings. The measurements from 24 ultrasound sensors (inputs) and the associated translational and rotational velocities (outputs) at a given sampling time were acquired during the execution of a Wall-Following navigation task. A total of 5456 points were collected, corresponding to approximately 4 clockwise rounds within the room where the experiment was realized. Once the data was collected, diferent architectures of the ELM(p,q,m) were trained using the first 4000 points (approximately 73% of the available data), where $p \in \{2, 4, 24\}$ denotes the dimension of the input vectors, $q \in \{5, 20, 50, 100, 200\}$ denotes the number of hidden neurons and $m \in \{1, 2\}$ denotes the number of output neurons. The remainder of the points was used to test the trained architectures. In the tests, the logistic sigmoid function was chosen as the activation function of the hidden neurons, while the output neurons were linear ones. Several tests were implemented to evaluate the performance of the ELM network in learning the translational and rotational velocities for different values of the parameters p (input vector dimension) and q (number of hidden neurons). As a general result, the ELM network was able to learn (i.e. to model) the input-output relation for the translational velocity, achieving the best result for $p = 2$ e $q = 200$. However, for the rotational velocity, the ELM network achieved a very poor result, not being able of learning the corresponding input-output mapping.

Keywords: ELM, robot SCITOS G5, ultrasound sensors, neural networks, Wall-Following.

Dedico este trabalho aos meus pais,
aos meus irmãos, à minha familia e aos amigos.

Agradecimentos

Ao meu Orientador Prof. Dr. Guilherme de Alencar Barreto pelo incentivo e atenção no auxílio às atividades e discussões sobre o andamento deste Trabalho de Conclusão de Curso. Aos demais coordenadores e funcionários da Universidade Federal do Ceará. A todos os professores e seus convidados pelo carinho, dedicação e entusiasmo demonstrado ao longo do ano. Aos meus amigos e colegas brasileiros de classe pela ajuda que sempre estavan dispostos a dar-me e a sorte de ter conhecido eles. A Irene pela compreensão, os momentos vividos e sua amizade que me foi dada e sei que é para a vida toda. E em especial, à minha família por me dar a oportunidade de desenvolver meu trabalho em um país que acabei amando, Brasil.

Nunca desista de um sonho. Apenas tente ver os sinais que os levam a ele.
Paulo Coelho

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

ELM *Extreme Learning Machine* (Maquina de Aprendizado Extremo)

RNA Rede Neural Artificial

RNAs Redes Neurais Artificiais

NUTEC Núcleo de Tecnologia Industrial do Ceará

Capítulo 1

Introdução

Com redes neurais procura-se a solução de problemas complexos, não como uma seqüência de passos, mas como a evolução dos sistemas de computador que inspirou o cérebro humano, e dotado de tanto "inteligência"de alguns, que são, mas a combinação de elementos simples de processamento (neurônios) interligados para operar simultaneamente. Tentaremos dotar dessa "inteligência"ao robô SCITOS G5, qual modelo vai ser utilizado no trabalho para a modelagem do robô móvel usando a rede neural ELM em um caso específico para as velocidades de rotação e translação.

Este capítulo vai apresentar no apartado 1.1 o contexto no qual está desenvolvido o projeto da rede, os objetivos marcados para realizar-lo no apartado 1.3. A motivação encontrada no apartado 1.4, e a importância da metodologia utilizada no problema da aplicação da rede neural no apartado 1.5. Finalmente vai concluir com a estrutura da monografia no apartado 1.6.

1.1 Contextualização

As redes neurais são apenas uma maneira de imitar determinadas características humanas, tais como a capacidade de memorizar e associar eventos. Se formos considerar cuidadosamente os problemas que não podem ser expressos através de um algoritmo, todos eles têm uma coisa em comum: a experiência. Uma maneira de abordar este problema é construir sistemas que sejam capazes de reproduzir esta característica humana.

Em sumo, as redes neurais são um modelo artificial e simplificado do cérebro humano, que é o exemplo mais perfeito que temos para um sistema que é capaz de adquirir conhecimento através da experiência. Uma rede neural é um novo sistema de processamento de informação, a unidade básica de processamento é inspirado na célula fundamental do sistema nervoso humano:. Neurônio.

Todos os processos do corpo humano estão relacionados de alguma forma ou de outra com a atividade desses neurônios. Elas são um componente relativamente

simples do ser humano, mas quando milhares deles estão ligados entre si são muito poderosos.

O fato de realizar uma modelagem das leituras sensoriais do robô móvel recopiladas numa trajetória feita no laboratório, com o fim de que o mesmo robô seja capaz de implementar a essa trajetória ele sozinho. Supõe um estudo específico e a implementação de memória das duas velocidades a modelar no robô.

1.2 Revisão bibliografica

Obter, projetar e construir máquinas capazes de executar processos com alguma inteligência tem sido um dos principais objetivos e preocupações dos cientistas ao longo da história. A idéia de poder aplicar inteligência neural aos sistemas computacionais foi sempre atrativa, entanto que representa um avanço para a humanidade.

Historicamente redes feed-forward foram introduzidas como modelos de redes neurais biológicas de acordo com McCulloche e Pitts em 1943. O recente desenvolvimento das redes feed-forward para aplicações de reconhecimento de padrões e aprendizagem, no entanto, continuou independentemente de considerações biológicas de modelagem.

Rumelhart redescobriu em 1986 o algoritmo de aprendizagem de propagação de volta (backpropagation) (AHALT S. C.; KRISHNAMURTHY,). Atualmente, existem muitos trabalhos que são realizados e publicados cada ano, as novas aplicações que surgem (especialmente na área de controle) e as empresas que lançam novos produtos para o mercado, tanto de hardware e software (especialmente para a simulação).

1.3 Objetivos

Esta secção vai descrever os objetivos a desenvolver com o uso da rede neural *Extreme Learning Machine* (Maquina de Aprendizado Extremo) (ELM) através de um panorama geral dos propósitos, os quais são que na possível implementação da rede ELM para a modelagem estática das leituras dos sensores das velocidades translacional e rotaciona do robô, esse possa concluir a aprendizagem feita ele sozinho.

1.3.1 Objetivo Geral

O principal objetivo deste trabalho é utilizar a rede neural ELM para gerar a aprendizagem do robô SCITOS G5 de um caminho determinado, baseando se nas medições das duas velocidades, recopilados pelos sensores ultrasom do robô feitas no Laboratório Núcleo de Tecnologia Industrial do Ceará (NUTEC) no campus do Pici da Universidade Federal do Ceará. Esses dados vão ser as entradas da rede, vai ter várias etapas nas que a rede realiza o controle de aprendizagem dos neurônios com o treinamento dos dados e posteriormente os testes de comprobação com os que . Na figura 1.1 está representada a velocidade translacional do robô que vamos modelar e na figura 1.2 a velocidade rotacional.

1.3.2 Objetivos Específicos

Para o desenvolvimento do trabalho alguns objetivos específicos foram necessários, eles estão listados a seguir:

i. **Aprendizagem do mapeamento das leituras sensoriais (entrada) na velocidade translacional (saída):** A rede vai ser capaz de aplicar o algoritmo de aprendizagem para gerar como saída a velocidade translacional do robô que vai governar o movimento das rotas.

ii. **Aprendizagem do mapeamento das leituras sensoriais (entrada) na velocidade rotacional (saída):** A rede vai ser capaz de aplicar o algoritmo de aprendizagem para gerar como saída a velocidade translacional do robô que vai governar a direção do movimento das rotas.

iii. **Desenvolvimento no ambiente Matlab:** Realização do algoritmo matemático da rede ELM, o código e a simulação no ambiente Matlab. O software permite a reprodução dos grafos para apresentar os resultados.

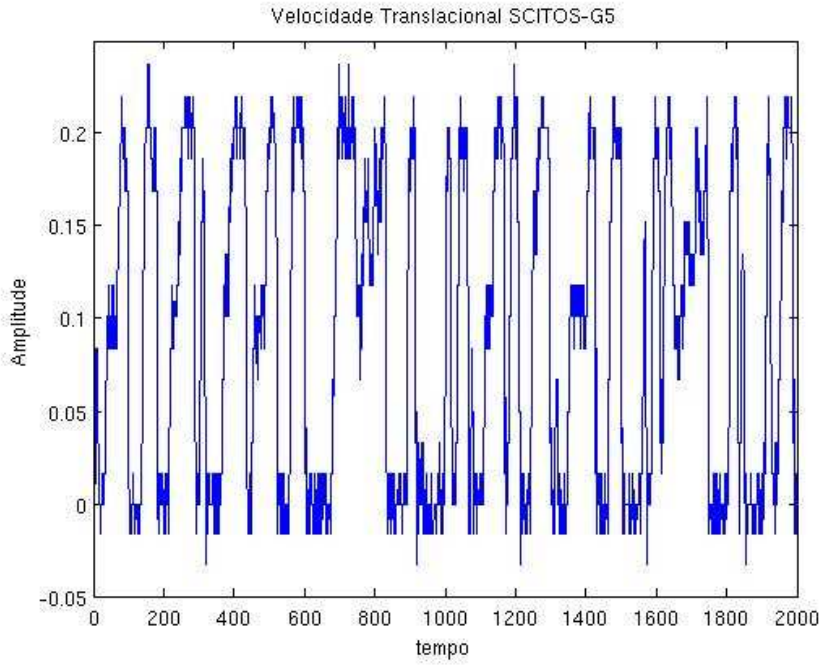

Figura 1.1: Velocidade Translacional.

1.4 Motivação

Existe agora uma forte tendência para estabelecer um novo campo da computação que integra os diferentes métodos de resolução de problemas que não podem ser descritas com facilidade usando abordagem algorítmica tradicional. Esses métodos têm a sua origem na emulação de sistemas biológicos. Esta é uma nova

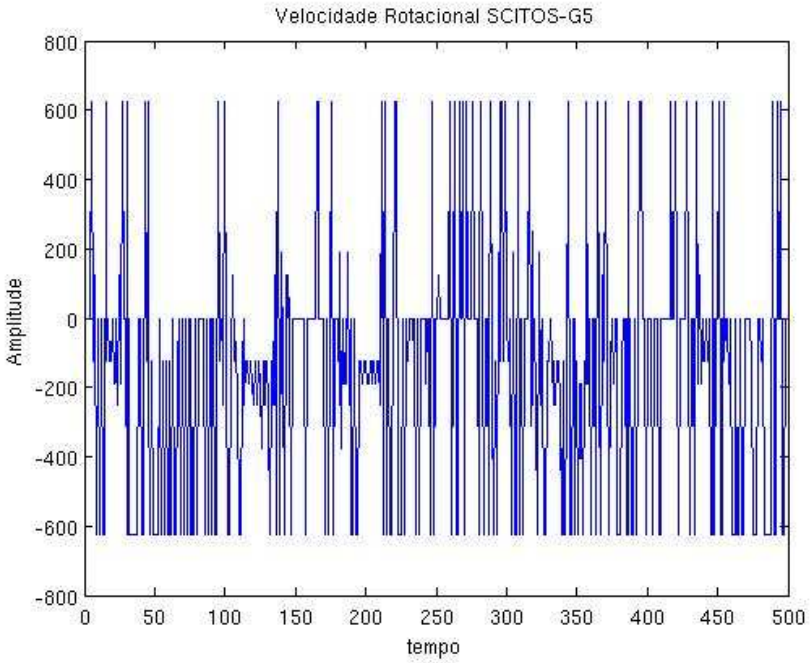

Figura 1.2: Velocidade Rotacional.

forma de computação que é capaz de lidar com a imprecisão e incertezas dos problemas relacionados com o mundo real (reconhecimento de padrões, tomada de decisão, etc). Isto terá a metodologia de redes neurais.

No trabalho vamos realizar um passo do que seria uma aprendizagem de um sistema físico, fazer-lo autônomo, e o fato de que se pode derivar num estudo muito mais específico como pode ser a implementação final no robo móvel. Fazemos a modelagem estática do mapeamento das leituras sensoriais com o objetivo de utilizar ela para a modelagem dinâmica e finalmente poder implementar no robô. Uma das motivações principais é o objetivo de fazer o robô SCITOS G5 autônomo na trajetória marcada.

Considerando que esta implantação representa o poder que separa os seres humanos de todo o resto do mundo ao seu redor, o conhecimento.

A capacidade adaptativa de aprendizagem é uma das características mais atrativas das redes neurais. Ou seja, aprender a executar tarefas mediante certo treinamento com exemplos ilustrativos. Como redes neurais podem aprender a distinguir padrões com treinamentos de exemplos, não é necessário desenvolver modelos ou necessidade de especificar em antecipação funções de distribuição de probabilidade. As redes neurais auto-adaptáveis são sistemas dinâmicos. Eles são resistentes devidos á capacidade de auto-ajustes dos elementos (neurônios) que compõem o sistema véase (KUNG, 1993).

1.5 Metodologia

A metodologia empregada na implementação da rede ELM pode ser descrita como um processo de varias etapas como é mostrado na figura 1.3.

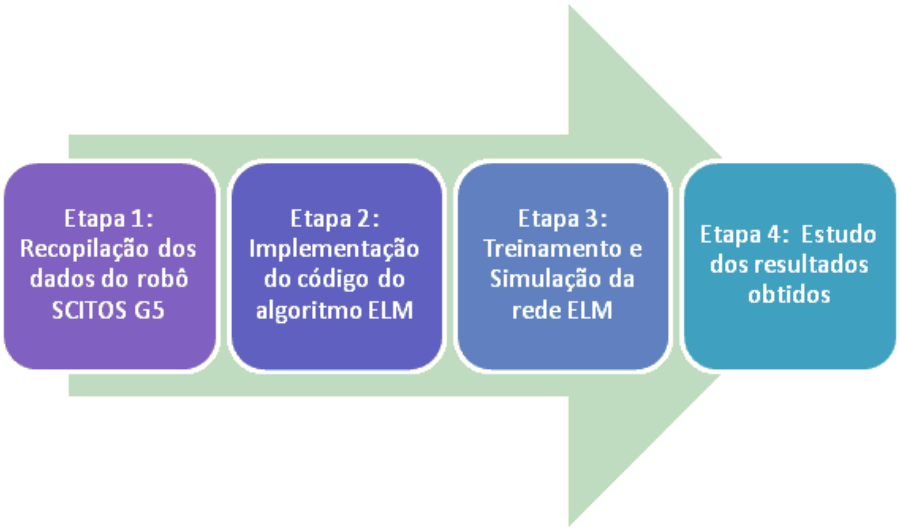

Figura 1.3: Etapas da metodologia utilizada.

Pode-se observar que a metodologia utilizada esta composta de quatro etapas principais usadas para a realização do trabalho, a primeira etapa é a recopilação dos dados dos sensores do robô SCITOS G5 feita no laboratório NUTEC da Universidade Federal do Ceará, o trabalho esta baseado nas 5456 medições. A segunda etapa é a codificação no ambiente Matlab do algoritmo da rede ELM com as dicas de (BARRETO, 2010), na terceira etapa se faz o treinamento e testes da rede ELM com os dados dos sensores, e na quarta etapa estudamos os resultados obtidos através dos gráficos das saídas e dos erros de cada teste simulados no Matlab.

1.6 Estrutura da Monografia

Este trabalho está organizado em 5 Capítulos. O segundo Capítulo realiza uma revisão sobre os fundamentos de Redes Neurais Artificiais (RNAs) e ELM. O Capítulo 3 descreve como as ferramentas serão usadas no desenvolvimento do trabalho, o ambiente de simulação, além de descrever o método utilizado para avaliar a eficiência de compactação da abordagem proposta. No Capítulo 4 são descritos e discutidos os resultados obtidos no trabalho e por fim, o Capítulo 5 apresenta as considerações finais e as perspectivas futuras.

Capítulo 2

Aplicação da Rede ELM ao Robô SCITOS G5

Neste capítulo, serão abordados os aspectos que definem a abordagem da rede neural ELM na aplicação ao Robô SCITOS G5.

2.1 Redes Neurais Artificiais. Conceitos Básicos

Na busca por máquinas inteligentes, um modelo cujo funcionamento se deseja replicar ou reproduzir é o cérebro humano. O cérebro humano possui características úteis para uso em qualquer sistema "inteligente"artificial, como por exemplo: robustez e tolerância a falhas; flexibilidade, adaptabilidade e capacidade de aprendizado; capacidade de processar informação nebulosa, probabilística, ruidosa ou inconsistente; processamento paralelo; tamanho reduzido, grande capacidade de processamento e baixo consumo de energia. Logo, é natural que surjam modelos computacionais que visam reproduzir algumas das "virtudes"listadas acima, e que tais modelos possam ser utilizados em máquinas, onde se espera que melhoria de seu desempenho durante a execução de uma tarefa específica.

Em geral, tais modelos artificiais exploram uma característica específica do funcionamento do cérebro (memória, visão, etc.), de modo que uma gama enorme de algoritmos tem sido propostos. Contudo, estes modelos guardam um grande número de semelhanças entre si, como por exemplo serem constituídos de modelos simplificados de células nervosas, também chamadas de neurônios.

O neurônio é o elemento construtivo básico do sistema nervoso. O cérebro humano é composto por aproximadamente 10^{11} a 10^{12} células nervosas, com cerca de 1015 interconexões em caminhos de transmissão. Os neurônios se ligam entre si formando uma imensa e complexa rede, na qual recebem impulsos eletroquímicos, processam estes impulsos e os retransmitem a outras células nervosas. São os neurônios e suas conexões sinápticas que servem de inspiração às redes neurais artificiais. (ROSEMBLATT, 1958) aponta cinco componentes importantes em um neurônio ver Figura 2.1.

- DENDRITOS: em forma de galhos secos, são captadores e condutores de estímulos nervosos, excitatórios ou inibitórios, de outros neurônios, transportandoos ao corpo celular.
- CORPO CELULAR: neste encontra-se o Núcleo, que é o responsável pelo processamento dos estímulos nervosos aferentes. Envia um novo impulso ao axônio, dependendo da comparação entre o resultado do processamento e a diferença de potencial entre as paredes interna e externa do neurônio.
- AXÔNIO: expansão linear do corpo celular, conduzindo informações a grandes distâncias, por propagação de um sinal elétrico transitório, chamado potencial de ação.
- SINAPSES: região situada entre as terminações do axônio de um neurônio (denominadas pré-sinápticas) e as superfícies receptoras dos dentritos ou do corpo celular de outro neurônio (denominadas pós-sinápticas). As sinapses funcionam como válvulas capazes de controlar a transmissão de impulsos nervosos ou potenciais de ação e regular a sua intensidade. Durante a vida de um sistema nervoso, as sinapses estão em constante formação e modificação

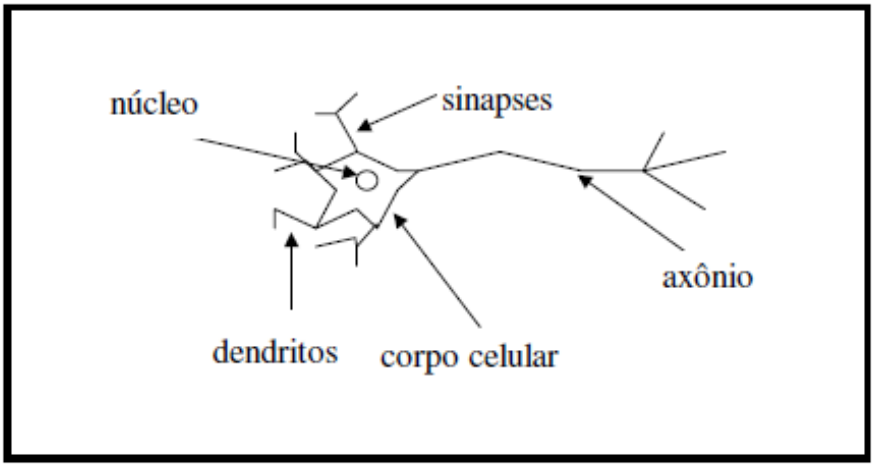

Figura 2.1: Esquema simplificado de um neurônio.

Assim, uma rede neural artificial Rede Neural Artificial (RNA) pode ser vista como um conjunto de elementos processadores simples, baseados em neurônios, que são ligados uns aos outros através de conexões análogas às sinapses. Estas conexões armazenam o "conhecimento"da rede e os diversos padrões de atividade expressam os vários objetos codificados pela rede.

Em outras palavras, as conexões fazem o papel de memória de longo prazo, enquanto que o estado de ativação das unidades de uma rede realizam o papel de memória de curto prazo. O conhecimento da rede é adquirido por meio de um processo de treinamento no qual, em suas versões mais básicas, apenas as conexões

entre as unidades são variadas através da mudanças de pesos. O entendimento do neurônio é, portanto, o ponto de partida para criação de uma rede neural.

O modelo mais explorado e que serve como base para a maioria dos estudos, é o PERCEPTRON que foi proposto originalmente por (ROSEMBLATT, 1958).

O Perceptron na figura 2.2 equivale a um único neurônio, tem sua forma e características inspiradas no neurônio biológico. É através desta abstração que se constróem diferentes modelos de redes neurais com características próprias, mas que possuem o Perceptron como unidade básica. As entradas equivalem a sinais que chegam aos dendritos, os pesos representam as sinapses, as regras de propagação e ativação são a função do núcleo e a saída representa o sinal que o axônio propaga.

As entradas $x_i, i = 1, 2, 3, ..., n$ são binárias; os pesos w_{ji} podem ser positivos (excitatórios) ou negativos (inibitórios); a regra de agregação é dada por: $NET_j = \sum_{i=0}^{n} w_{ji}x_i$, a regra de ativação determina se o somatório é maior ou não que um certo limiar (threshold); a saída será 1 se a somatória for maior ou igual ao limiar, e será 0 se a somatória for menor que o limiar. O conhecimento do sistema é armazenado pelas conexões. O treinamento do sistema consiste em alterar os valores destas conexões, empregando algoritmos de treinamento, até obtermos uma saída desejada.

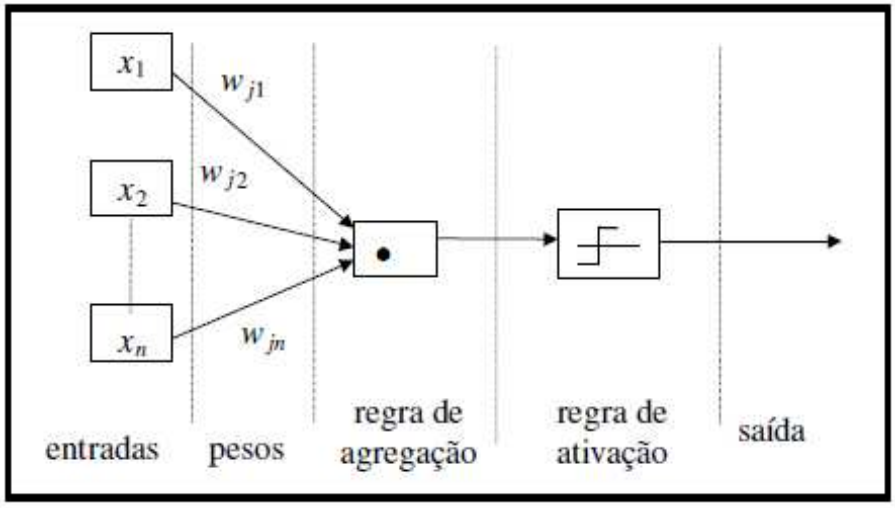

Figura 2.2: Modelo chamado PERCEPTRON.

Algoritmos de treinamento de redes neurais podem ser supervisionados ou nãosupervisionados. No primeiro caso, utiliza-se um "professor". Durante o treinamento, são fornecidos pares formados pela entrada e pela saída-desejada e apenas as conexões entre as unidades são variadas através das mudanças dos pesos ajustados como função do erro entre a saída desejada e a saída obtida pela rede em treinamento.

No caso dos algoritmos não-supervisionados, a própria rede se encarrega de fazer o mapeamento entre um espaço de entrada para um espaço de saída,

auto-organizando suas conexões sinápticas para isto. Os algoritmos supervisionados são normalmente mais usados em aplicações de Engenharia, por encontrarem uma resposta desejada. Contudo, os modelos não-supervisionados são mais plausíveis sob os pontos de vista biológico e psicológico.

2.1.1 *Framework* para Análise e Projeto de RNAs

No livro (MCCLELLAND, 1986) propuseram oito características, comuns à maior parte dos modelos de redes neurais artificiais, as quais formam um framework básico discutido a seguir:

i. Um conjunto de unidades de processamento: é o primeiro estágio da elaboração de um modelo de RNAs. Possui dois aspectos importantes: o primeiro é quanto cada unidade de processamento vai se parecer com uma célula neural; o segundo é o significado de cada unidade, ou seja, o que ela representa individualmente e em conjunto.

ii. Um padrão de conectividade entre as diferentes unidades: O padrão de conectividade define o conhecimento do sistema, determinando como uma rede responde a uma dada entrada. Tal padrão é codificado por pesos w_{ji}, representando conexões sinápticas excitatórias ou inibitórias entre as unidades i e j

iii. Definição de diferentes estados de ativação: cada estado de ativação de uma unidade em uma rede neural tem um significado associado. Existem inúmeras possibilidades de estados de ativação das unidades e suas escolhas levam em conta a representação escolhida. O estado de ativação pode ser: contínuo ou discreto; limitado ou ilimitado; ter valores reais, binários, bipolares, ou série de valores.

iv. Uma Regra de Propagação: A função desta regra é combinar os valores de entrada de uma unidade com os valores da matriz de conexões para produzir o efeito total "sentido"pela unidade. A regra de propagação equivale à modificação da situação eletroquímica do neurônio.

v. Uma regra de ativação: Esta regra atualiza o estado de ativação de cada unidade, considerando-se a entrada ponderada desta unidade e seu estado de ativação presente. A regra determina se a unidade está ativa ou não.

vi. Uma função de saída para cada unidade: Os sinais que fazem com que as unidades interajam são transmitidos por uma função de saída $y_j = (f_{aj}(t), w_{ji}(t), t)$. A função de saída transforma o estado de ativação de uma unidade $a_j(t)$ para um sinal de saída $y_j(t)$.

vii. Uma regra de aprendizagem: Mudanças nos pesos (conexões) sinápticos são coordenadas por um processo de aprendizagem que pode ser envolver três situações: aparecimento de novas conexões; perda de conexões existentes e modificação de conexões já existentes.

viii. Um ambiente no qual o sistema opera: A definição do ambiente deve considerar: sinais de entrada, que são fornecidos à rede como características do meio; um contexto, representando na rede informações importantes para o processamento das entradas, e uma saída que leva informação processada ao meio.

Definidas as características de um modelo de rede neural tais como: o número de unidades de processamento nas camadas diversas camadas, as regras de ativação e propagação, a regra de aprendizagem, os tipos de conexões entre unidades da mesma camada e entre diferentes camadas e também o meio em que vai atuar, este modelo está apto a ser testado em diferentes áreas

2.2 Rede ELM

Nas seguintes líneas descrevemos tudos os passos e considerações feitas para a implementação da rede ELM do documento (BARRETO, 2010).

2.2.1 Definições Preliminares

De início, vamos assumir que existe uma lei matemática $\mathbf{F}(\cdot)$, também chamada aqui de função ou mapeamento, que relaciona um vetor de entrada qualquer, $\mathbf{x} \in \mathbb{R}^{p+1}$, com um vetor de saída, $\mathbf{d} \in \mathbb{R}^m$. Esta relação, representada genericamente na Figura 2.3, pode ser descrita matematicamente da seguinte forma:

$$\mathbf{d} = \mathbf{F}[\mathbf{x}] \tag{2.1}$$

em que se assume que $\mathbf{F}(\cdot)$ é totalmente desconhecida, ou seja, não sabemos de antemão quais são as *fórmulas* usadas para associar um vetor de entrada $\mathbf{x}$ com seu vetor de saída $\mathbf{d}$ correspondente.

O mapeamento $\mathbf{F}(\cdot)$ pode ser tão simples quanto um mapeamento linear, tal como

$$\mathbf{d} = \mathbf{M}\mathbf{x} \tag{2.2}$$

em que $\mathbf{M}$ é uma matriz de dimensão $(p+1) \times m$. Contudo, $\mathbf{F}(\cdot)$ pode ser bastante complexo, envolvendo relações não-lineares entre as variáveis de entrada e saída. É justamente o funcionamento da relação matemática $\mathbf{F}(\cdot)$ que se deseja *imitar* através do uso de algoritmos adaptativos, tais como as redes neurais.

Supondo que a única fonte de informação que nós temos a respeito de $\mathbf{F}(\cdot)$ é conjunto finito de N pares entrada-saída observados (ou medidos), ou seja:

$$\begin{matrix} \mathbf{x}_1, & \mathbf{d}_1 \\ \mathbf{x}_2, & \mathbf{d}_2 \\ \vdots & \vdots \\ \mathbf{x}_N, & \mathbf{d}_N \end{matrix} \tag{2.3}$$

Figura 2.3: Representação simplificada de um mapeamento entrada-saída genérico.

Os pares entrada-saída mostrados acima podem ser representados de maneira simplificada como $\{\mathbf{x}_\mu, \mathbf{d}_\mu\}$, em que μ é um apenas indíce simbolizando o μ-ésimo par do conjunto de dados. Uma maneira de se adquirir conhecimento sobre $\mathbf{F}(\cdot)$ se dá exatamente através dos uso destes pares.

Para isto pode-se utilizar uma rede neural qualquer para implementar um mapeamento entrada-saída aproximado, representado como $\hat{\mathbf{F}}(\cdot)$, tal que:

$$\mathbf{y}_\mu = \hat{\mathbf{F}}[\mathbf{x}_\mu] \tag{2.4}$$

em que $\mathbf{y}_\mu$ é a saída gerada pela rede neural em resposta ao vetor de entrada $\mathbf{x}_\mu$. Esta saída, espera-se, seja muito próxima da saída real $\mathbf{d}_\mu$. Dá-se o nome de *Aprendizado Indutivo* ao processo de obtenção da relação matemática geral $\hat{\mathbf{F}}(\cdot)$ a partir de apenas alguns pares $\{\mathbf{x}_\mu, \mathbf{d}_\mu\}$ disponíveis.

A seguir será mostrado uma arquitetura de redes neurais recentemente proposta com o propósito de obter uma representação aproximada de um mapeamento entrada-saída genérico.

2.2.2 Máquina de Aprendizado Extremo

Estamos considerando nas definições e cálculos a seguir uma arquitetura de rede neural do tipo *feedforward* (i.e. sem realimentação) com apenas uma camada de neurônios ocultos, conhecida como Máquina de Aprendizado Extremo (*Extreme Learning Machine*, ELM. Esta arquitetura de rede neural é semelhante à rede MLP, porém apresenta uma fase de aprendizado infinitamente mais rápida que a da rede MLP. Começaremos a seguir com a descrição de sua arquitetura, para em seguida escrever sobre o funcionamento e o treinamento da rede ELM.

Os neurônios da camada oculta (primeira camada de pesos sinápticos) são

representados conforme mostrado na Figura 2.4a, enquanto os neurônios da camada de saída segunda camada de pesos sinápticos) são representados conforme mostrado na Figura 2.4b.

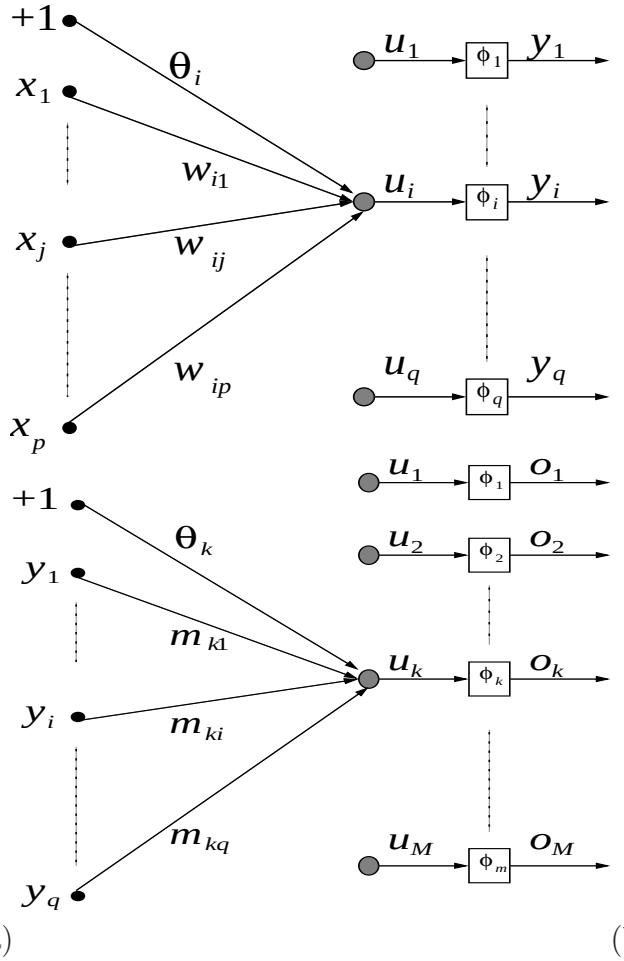

Figura 2.4: (a) Neurônio da camada escondida. (b) Neurônio da camada de saída.

O vetor de pesos associado a cada neurônio i da camada escondida, também chamada de *camada oculta* ou *camada intermediária*, é representado como

$$\mathbf{w}_i = \begin{pmatrix} w_{i0} \\ \vdots \\ w_{ip} \end{pmatrix} = \begin{pmatrix} \theta_i \\ \vdots \\ w_{ip} \end{pmatrix} \tag{2.5}$$

em que θ_i é o limiar (*bias* ou *threshold*) associado ao neurônio i. Os neurônios desta camada são chamados de neurônios escondidos por não terem acesso direto à sáida da rede, onde são calculados os erros de aproximação.

De modo semelhante, o vetor de pesos associado a cada neurônio k da camada

de saída é representado como

$$\mathbf{m}_k = \begin{pmatrix} m_{k0} \\ \vdots \\ m_{kq} \end{pmatrix} = \begin{pmatrix} \theta_k \\ \vdots \\ m_{kq} \end{pmatrix} \tag{2.6}$$

em que θ_k é o limiar associado ao neurônio de saída k.

O treinamento da rede ELM se dá em duas etapas, que são descritas a seguir.

2.2.3 Fase 1: Inicialização Aleatória dos Pesos dos Neurônios Ocultos

Esta etapa de funcionamento da rede ELM envolve o cálculo das ativações e saídas de todos os neurônios da camada escondida e de todos os neurônios da camada de saída, uma vez que os pesos w_{ij}, $i = 1, \ldots, q$ e $j = 0, \ldots, p$, tenham sido inicialidos com valores aleatórios. Formalmente, podemos escrever:

$$w_{ij} \sim U(a, b) \qquad \text{ou} \qquad w_{ij} \sim N(0, \sigma^2) \tag{2.7}$$

em que $U(a, b)$ é um número (pseudo-)aleatório uniformemente distribuído no intervalo (a, b), enquanto $N(0, \sigma^2)$ é um número (pseudo-)aleatório normalmente distribuído com média zero e variância σ^2.

Em ambientes de programação tais Matlab© e Octave, esta fase é facilmente implementada em uma linha apenas de código. Para isso, precisamos definir uma matriz de pesos $\mathbf{W}$, com q linhas e $p + 1$ colunas:

$$\mathbf{W} = \begin{pmatrix} w_{10} & w_{11} & \cdots & w_{1p} \\ w_{20} & w_{21} & \cdots & w_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ w_{q0} & w_{q1} & \cdots & w_{qp} \end{pmatrix}_{q \times (p+1)} = \begin{pmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \vdots \\ \mathbf{w}_q^T \end{pmatrix} \tag{2.8}$$

em que notamos que i-ésima linha da matrix $\mathbf{W}$ é composta pelo vetor de pesos do i-ésimo neurônio oculto.

2.2.4 Fase 2: Acúmulo das Saídas dos Neurônios Ocultos

Em esta fase do treinamento da rede o fluxo de sinais (informação) se dá dos neurônios de entrada para os neurônios de saída, passando obviamente pelos neurônios da camada escondida. Por isso, diz-se que o informação está fluindo no sentido **direto** (*forward*), ou seja:

Entrada → Camada Intermediária → Camada de Saída

Assim, após a apresentação de um vetor de entrada $\mathbf{x}$, na iteração t, o primeiro passo é calcular as ativações dos neurônios da camada escondida:

$$u_i(t) = \sum_{j=0}^{p} w_{ij} x_j(t) = \mathbf{w}_i^T \mathbf{x}(t), \quad i = 1, \ldots, q \tag{2.9}$$

em que T indica o vetor (ou matriz) transposto e q indica o número de neurônios da camada escondida.

A operação seqüencial da Eq. (2.9) pode ser feita de uma única vez se utilizarmos a notação vetor-matriz. Esta notação é particularmente útil em ambinetes do tipo Matlab/Octave. Neste caso, temos que o vetor de ativações $\mathbf{u}_i(t) \in \mathbb{R}^q$ do i-ésimo neurônio oculto na iteração t é calculado como

$$\mathbf{u}(t) = \mathbf{W}\mathbf{x}(t). \tag{2.10}$$

Em seguida, as saídas correspondentes são calculadas como

$$z_i(t) = \phi_i(u_i(t)) = \phi_i \left(\sum_{j=0}^{p} w_{ij}(t) x_j(t) \right) = \phi_i \left(\mathbf{w}_i^T(t) \mathbf{x}(t) \right) \tag{2.11}$$

tal que a função de ativação ϕ assume geralmente uma das seguintes formas:

$$\phi_i(u_i(t)) = \frac{1}{1 + \exp[-u_i(t)]}, \qquad \text{(Logística)} \tag{2.12}$$

$$\phi_i(u_i(t)) = \frac{1 - \exp[-u_i(t)]}{1 + \exp[-u_i(t)]}, \qquad \text{(Tangente Hiperbólica)} \tag{2.13}$$

No nosso trabalho se utiliza a Logística.

Em notação matriz-vetor, a Eq. (2.11) pode ser escrita como

$$\mathbf{z}(t) = \phi_i(\mathbf{u}_i(t)) = \phi_i(\mathbf{W}\mathbf{x}(t)). \tag{2.14}$$

em que a função de ativação $\phi_i(\cdot)$ é aplicada a cada um dos q componente do vetor $\mathbf{u}(t)$.

Para cada vetor de entrada $\mathbf{x}(t)$, $t = 1, \ldots, N$, tem-se um vetor $\mathbf{z}(t)$ correspondente, que deve ser organizado (disposto) como uma coluna de uma matriz $\mathbf{Z}$. Esta matriz terá q linhas por N colunas:

$$\mathbf{Z} = [\mathbf{z}(1) \mid \mathbf{z}(2) \mid \cdots \mid \mathbf{z}(N)]. \tag{2.15}$$

A matriz $\mathbf{Z}$ será usada na Fase 3 para calcular os valores do pesos dos neurônios de saída da rede ELM.

2.2.5 Fase 3: Cálculo dos Pesos dos Neurônios de Saída

Sabemos que para cada vetor de entrada $\mathbf{x}(t)$, $t = 1, \ldots, N$, tem-se um vetor de saídas desejadas $\mathbf{d}(t)$ correspondente. Se organizamos estes N vetores ao longo das colunas de uma matriz $\mathbf{D}$, então temos que esta matriz terá dimensão m linhas e N colunas:

$$\mathbf{D} = [\mathbf{d}(1) \mid \mathbf{d}(2) \mid \cdots \mid \mathbf{d}(N)]. \tag{2.16}$$

Podemos entender o cálculo dos pesos da camada de saída como o cálculo dos parâmetros de um mapeamento linear entre a camada oculta e a camada de saída. O papel de vetor de entrada para a camada de saída na iteração t é desempenhado pelo vetor $\mathbf{z}(t)$ enquanto o vetor de saída é representado pelo vetor $\mathbf{d}(t)$. Assim, buscamos determinar a matriz $\mathbf{M}$ que melhor represente a transformação

$$\mathbf{d}(t) = \mathbf{M}\mathbf{z}(t). \tag{2.17}$$

Para isso, podemos usar o método dos mínimos quadrados, também conhecido como método da pseudoinversa, já discutido nas notas de aula sobre o classificador OLAM. Assim, usando as matrizes $\mathbf{Z}$ e $\mathbf{D}$, a matriz de pesos $\mathbf{M}$ é calculada por meio da seguinte expressão:

$$\mathbf{M} = \mathbf{D}\mathbf{Z}^T \left(\mathbf{Z}\mathbf{Z}^T\right)^{-1}. \tag{2.18}$$

Alguns comentários sobre a matriz $\mathbf{M}$ fazem-se necessários:

- Note que para satisfazer a Eq. (2.17) a matriz $\mathbf{M}$ tem dimensão $m \times q$.
- A k-ésima linha da matriz $\mathbf{M}$, denotado aqui por $\mathbf{m}_k$, $k = 1, 2, \ldots, m$, corresponde ao vetor de pesos do k-ésimo neurônio de saída.

2.2.6 Teste e Capacidade de Generalização da Rede ELM

Uma vez determinadas as matrizes de pesos $\mathbf{W}$ e $\mathbf{M}$ temos a rede ELM pronta para uso. Durante o uso (teste) da rede ELM, calculamos as ativações dos neurônios da camada de saída por meio da seguinte expressão:

$$a_k(t) = \sum_{i=0}^{q} m_{ki}(t) z_i(t) = \mathbf{m}_k^T \mathbf{z}(t), \quad k = 1, \ldots, m \tag{2.19}$$

em que m é o número de neurônios de saída. Note que as saídas dos neurônios da camada oculta, $z_i(t)$, fazem o papel de entrada para os neurônios da camada de saída.

Em notação vetor-matriz, as operações da Eq. (2.19) podem ser executados de uma só vez por meio da seguinte expressão:

$$\mathbf{a}(t) = \mathbf{M}\mathbf{z}(t). \tag{2.20}$$

Para a rede ELM, assumimos que os neurônios de saída usam a função identidade como função de ativação, ou seja, as saídas destes neurônios são iguais às suas ativações: calculadas como:

$$y_k(t) = \phi_k(a_k(t)) = a_k(t). \tag{2.21}$$

Por generalização adequada entende-se a habilidade da rede em utilizar o conhecimento armazenado nos seus pesos e limiares para gerar saídas coerentes para novos vetores de entrada, ou seja, vetores que não foram utilizados durante o treinamento. A generalização é considerada boa quando a rede, durante o treinamento, foi capaz de capturar (aprender) adequadamente a relação entrada-saída do mapeamento de interesse.

O bom treinamento de uma rede ELM, de modo que a mesma seja capaz de lidar com novos vetores de entrada, depende de uma série de fatores, dentre os quais podemos listar os seguintes

i. Excesso de graus de liberdade de uma rede ELM, na forma de elevado número de parâmetros ajustáveis (pesos e limiares).

ii. Excesso de parâmetros de treinamento, tais como taxa de aprendizagem, fator de momento, número de camadas ocultas, critério de parada, dimensão da entrada, dimensão da saída, método de treinamento, separação dos conjuntos de treinamento e teste na proporção adequada, critério de validação, dentre outros.

Em particular, no que tange ao número de parâmetros ajustáveis, uma das principais conseqüências de um treinamento inadequado é a ocorrência de um subdimensionamento ou sobredimensionamento da rede ELM, o que pode levar, respectivamente, à ocorrência de *underfitting* (subajustamento) ou *overfitting* (sobreajustamento) da rede aos dados de treinamento. Em ambos os casos, a capacidade de generalização é ruim.

Dito de maneira simples, o subajuste da rede aos dados ocorre quando a rede não tem poder computacional (i.e. neurônios na camada oculta) suficiente para aprender o mapeamento de interesse. No ouro extremo está o sobreajuste, que ocorre quando a rede tem neurônios ocultos demais (dispostos em uma ou duas camadas ocultas) e passar a memorizar os dados de treinamento. O ajuste ideal é obtido para um número de camadas ocultas e neurônios nestas camadas que confere à rede um bom desempenho durante a fase de teste, quando sua generalização é avaliada.

2.2.7 Dicas para um Bom Desempenho da Rede ELM

O projeto de uma rede neural envolve a especificação de diversos itens, cujos valores influenciam consideravelmente funcionamento do algoritmo. A seguir especificaremos a lista destes itens juntamente com as faixas de valores que os mesmos podem assumir:

Dimensão do vetor de Entrada (p): Este item pode assumir em tese valores entre 1 e ∞. Porém, existe um limite superior que depende da aplicação de interesse e do custo de se medir (observar) as variáveis x_j. É importante ter em mente que um valor alto para p não indica necessariamente um melhor desempenho para a rede neural, pois pode haver redundância no processo de medição. Neste caso, uma certa medida é, na verdade, a combinação linear de outras medidas, podendo ser descartada sem prejuízo ao desempenho da rede. Quando é muito caro, ou até impossível, medir um elevado número de variáveis x_j, deve-se escolher aquelas que o especialista da área considera como mais relevante ou representativas para o problema. O ideal seria que cada variável x_j, $j = 1, \ldots, p$, "carregasse" informação que somente ela contivesse. Do ponto de vista estatístico, isto equivale a dizer que as variáveis são *independentes* ou *não-correlacionadas* entre si.

Dimensão do vetor de saída (M): Assim como o primeiro item, este também depende da aplicação. Se o interesse está em problemas de aproximação de funções, $\mathbf{y} = F(\mathbf{x})$, o número de neurônios deve refletir diretamente a quantidades de funções de saída desejadas (ou seja, a dimensão de $\mathbf{y}$).

Se o interesse está em problemas de classificação de padrões, a coisa muda um pouco de figura. Neste caso, o número de neurônios deve codificar o número de classes desejadas.

É importante perceber que estamos chamando as classes às quais pertencem os vetores de dados de uma forma bastante genérica: classe 1, classe 2, ..., etc. Contudo, à cada classe pode estar associado um rótulo (e.g. classe dos empregados, classe dos desempregados, classe dos trabalhadores informais, etc.), cujo significado depende da interpretação que o especialista na aplicação dá a cada uma delas. Estes rótulos normalmente não estão na forma numérica, de modo que para serem utilizados para treinar a rede ELM eles devem ser convertidos apra a forma numérica. A este procedimento dá-se o nome de codificação da saída da rede.

A codificação mais comum define como vetor de saídas desejadas um vetor binário de comprimento unitário; ou seja, apenas uma componente deste vetor terá o valor "1", enquanto as outras terão o valor "0" (ou -1). A dimensão do vetor de saídas desejadas corresponde ao número de classes do problema em questão. Usando esta codificação define-se automaticamente um neurônio de saída para cada classe. Por exemplo, se existem três classes possíveis, existirão três neurônios de saída, cada um representando uma classe. Como um vetor de entrada não pode pertencer a mais de uma classe ao mesmo tempo, o vetor de saídas desejadas terá valor 1 (um) na componente correspondente à classe deste vetor, e 0 (ou -1) para as outras componentes. Por exemplo, se o vetor de entrada $\mathbf{x}(t)$ pertence à classe 1, então seu vetor de saídas desejadas é $\mathbf{d}(t) = [1 \quad 0 \quad 0]^T$. Se o vetor $\mathbf{x}(t)$ pertence à classe 2, então seu vetor de saídas desejadas é $\mathbf{d}(t) = [0 \quad 1 \quad 0]^T$ e assim por diante para cada exemplo de treinamento.

Número de neurônios na camada escondida (q): Encontrar o número ideal de neurônios da camada escondida não é uma tarefa fácil porque depende de uma série de fatores, muito dos quais não temos controle total. Entre os fatores mais importantes podemos destacar os seguintes:

i. Quantidade de dados disponíveis para treinar e testar a rede.

ii. Qualidade dos dados disponíveis (ruidosos, com elementos faltantes, etc.)

iii. Número de parâmetros ajustáveis (pesos e limiares) da rede.

iv. Nível de complexidade do problema (não-linear, discontínuo, etc.).

O valor de q é geralmente encontrado por tentativa-e-erro, em função da capacidade de *generalização* da rede (ver definição logo abaixo). Grosso modo, esta propriedade avalia o desempenho da rede neural ante situações não-previstas, ou seja, que resposta ela dá quando novos dados de entrada forem apresentados. Se muitos neurônios existirem na camada escondida, o desempenho será muito bom para os dados de treinamento, mas tende a ser ruim para os novos dados. Se existirem poucos neurônios, o desempenho será ruim também para os dados de treinamento. O valor ideal é aquele que permite atingir as especificações de desempenho adequadas tanto para os dados de treinamento, quanto para os novos dados.

Existem algumas fórmulas heurísticas (*ad hoc*) que sugerem valores para o número de neurônios na camada escondida da rede ELM, porém estas regras devem ser usadas apenas para dar um valor inicial para q. O projetista deve sempre treinar e testar várias vezes uma dada rede ELM para diferentes valores de q, a fim de se certificar que a rede neural generaliza bem para dados novos, ou seja, não usados durante a fase de treinamento.

Dentre a regras heurísticas citamos a seguir três, que são comumente encontradas na literatura especializada:

Regra do valor médio - De acordo com esta fórmula o número de neurônios da camada escondida é igual ao valor médio do número de entradas e o número de saídas da rede, ou seja:

$$q = \frac{p + M}{2} \tag{2.22}$$

Regra da raiz quadrada - De acordo com esta fórmula o número de neurônios da camada escondida é igual a raiz quadrada do produto do número de entradas pelo número de saídas da rede, ou seja:

$$q = \sqrt{p \cdot M} \tag{2.23}$$

Regra de Kolmogorov De acordo com esta fórmula o número de neurônios da camada escondida é igual a duas vezes o número de entradas da rede adicionado de 1, ou seja:

$$q = 2p + 1 \tag{2.24}$$

Perceba que as regras só levam em consideração características da rede em si, como número de entradas e número de saídas, desprezando informações úteis, tais como número de dados disponíveis para treinar/testar a rede e o erro de generalização máximo aceitável.

Uma regra que define um valor inferior para q levando em consideração o número de dados de treinamento/teste é dada por:

$$q \geq \frac{N-1}{p+2} \tag{2.25}$$

A regra geral que se deve sempre ter em mente é a seguinte: *devemos sempre ter muito mais dados que parâmetros ajustáveis*. Assim, se o número total de parâmetros (pesos + limiares) da rede é dado por $Z = (p+1)\cdot q + (q+1)\cdot M$, então devemos sempre tentar obedecer à seguinte relação:

$$N \gg Z \tag{2.26}$$

Capítulo 3

Desenho da rede ELM aplicada ao robô SCITOS G5

Neste capítulo fazemos uma breve descrição da rede ELM aplicada ao problema da modelagem do robô SCITOS G5 e a determinação das entradas e saídas da rede.

3.1 Robô móvel SCITOS G5

Desenvolvido pela equipe da empresa MetraLabs Robotics localizada em Ilmenau, Alemanha, o robô SCITOS G5 é uma plataforma mecatrônica móvel profissional para pesquisa e desenvolvimento que combina as vantagens de robôs industriais, tais como robustez e longevidade, com a mobilidade e flexibilidade de um robô de pesquisa. A Figura 3.1 mostra o robô no laboratório de Robótica (LABOR), laboratório integrante do Centro de Referência em Automação e Robótica (CENTAURO), instalado nas dependências da Universidade Federal do Ceará, onde foram realizados os testes de navegação.

Nas seções seguintes estão detalhados o hardware e as bibliotecas de software disponíveiscom esta plataforma.

3.1.1 O Hardware

O robô SCITOS G5 possui vários sensores diferentes, estações auto-recarregáveis e interface homem-máquina para aplicações em ambientes fechados, com a possibilidade de mudar algumas outras interfaces (mecânicas e elétricas) de acordo com a necessidade do usuário. O robô move-se através de um sistema motor diferencial, que permite movimentar a plataforma de 60kg a uma velocidade de até 1,4m/s, capaz de rotacionar 360 graus e empurrar cargas de até 50kg.

O robô SCITOS G5 tem 570mm x 735mm x 610mm de altura, comprimento e largura, respectivamente. Possui ainda uma autonomia de 12h com um

Figura 3.1: Robô SCITOS G5 no Laboratório de Robótica (LABOR/CENTAURO).

PC embarcado. Este último é constituído de uma placa-mãe Mini-ITX, processador Intel Core Duo 1,6 ou 2,0GHz, com memoria RAM de 2GB, HD de 120GB, com entrada PS/2 para teclado e mouse, 5 entradas USB, 3 entradas Firewire2, TV-out, RS232, VGA, saída SPDIF, LVDS, e entrada RJ-45 para Ethernet 10/100, além de rede wireless on-board, padrão IEEE 802.11a/b/g. Como sistema operacional, utiliza Linux Fedora Core 8, com MetraLabs C++ Robot-API.

O robô possui encoders para cálculo de posição com 460 "ticks"por rotação da roda, sensor de colisão e 24 sensores de ultra-som com alcance de 20cm-300cm. Além desses sensores, há ainda um sensor a laser (Hokuyo URG-04LX, SICK S300 or LMS-200), uma cabeça robótica, câmera omnidirecional e portas de I/O digitais e analógicas.

Para interface humano-máquina o robô possui um monitor touch screen de 15 polegadas com resolução de 1024x768 pixels. Também possui 2 caixas de som embutidas e 2 microfones omnidirecionais. Além disso, uma cabeça robótica envolta por uma redoma de acrílico transparente com 300mm de diâmetro, na qual estão inseridos um par de olhos falsos e pálpebras móveis. Os movimentos que a cabeça robótica é capaz de realizar são:

- Elevação e abaixamento da cabeça ($20\circ, -7\circ$);
- Rotação de toda a cabeça em até ($350\circ$);
- Os movimentos sincronizados dos olhos para cima, para baixo e para os lados;

- O abertura e fechamento das pálpebras de forma independente uma da outra.

3.1.2 API do SCITOS G5

Nesta seção é apresentada a API MLRobotics disponibilizada para leitura de dados sensoriais e envio de comandos de controle (setpoints) relacionados ao robô SCITOS G5. A MLRobotics é uma coleção de bibliotecas escritas em C++ e que permitem o desenvolvimento de programas para robôs móveis produzidos pela empresa MetraLabs. A maioria dessas bibliotecas dependem dos pacotes básicos MetraLabsBase e MetraLabsMath. O pacote MetraLabsBase consiste em algumas classes genéricas que podem ser utilizadas para operações gráficas, operações de leitura e escrita em arquivos comuns e XML, comunicação via socket, funções de criptografia e segurança, entre outros.

3.1.3 Metodologia da coleta dos dados

Para coletar dados destinados ao treinamento das redes neurais, foi criada uma rotina em C++ para guiar e rotular o comportamento de seguir paredes, dadas as leituras sensoriais naquele instante. A rotina, por ser a responsável por gerar as decisões que o robô precisa tomar sobre a sua movimentação, é colocada dentro de uma thread que é chamada a cada vez que os sensores de ultra-som captam alguma mudança significativa. Caso não haja nenhuma atividade sensorial em 500ms, o robô pára.

A lógica do algoritmo de navegação faz uso de regras heurísticas baseadas em condições pré-estabelecidas, as quais foram determinadas a partir de diversas tentativas e erros até que chegasse a um comportamento adequado ao problema. Ela atua sobre as medidas de distância captadas à frente e à esquerda do robô, embora este também calcule as distâncias referentes ao seu lado direito e atrás.

O vetor de leituras sensoriais no instante n, $x_s(n)$ é representado por:

$$x_s(n) = [x_f(n) \;\; x_e(n) \;\; x_d(n) \;\; x_t(n)]^T \tag{3.1}$$

em que os índices f, e, det indicam, respectivamente, a leitura correspondente aos sensores localizados na parte da frente, à esquerda, à direita e atrás. Cada uma dessas distâncias é extraída de um conjunto de sensores que formam juntos um feixe de 60◦, e a distância utilizada é a menor encontrada pelos sensores que fazem parte do conjunto. Utilizando a Figura 3.3 para melhor entender o conceito, a leitura referente à frente , $x_f(n)$, é dada por

$$x_f(n) = min\{x_0(n), x_1(n), x_{11}(n)\} \tag{3.2}$$

de forma similar, a leitura referente ao lado esquerdo pode ser representada como

$$x_e(n) = min\{x_8(n), x_9, (n)x_{10}(n)\} \tag{3.3}$$

As quatro distâncias são doravante referidas ao longo do texto como distâncias resumidas, e a Figura 3.2 mostra de forma intuitiva como são escolhidos os valores que vão representá-las em cada lado. Os números em vermelho, representam os sensores que obtêm a menor distância do seu grupo, e conseqüentemente são os escolhidos para representá-los. Essa abordagem também é a mesma utilizada no trabalho de (KRISHNA, 2000).

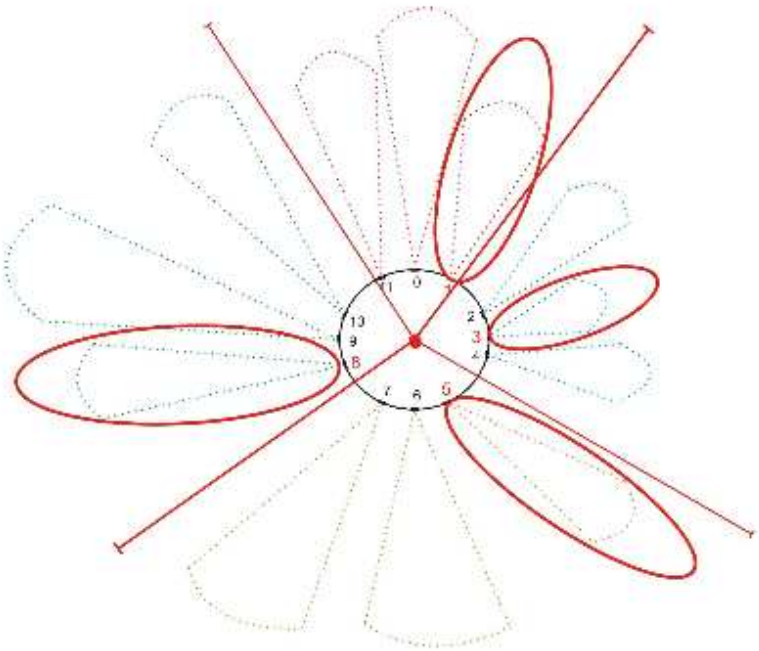

Figura 3.2: Ilustração das distâncias resumidas.

Além da lógica que determina a movimentação, a thread também possui a responsabilidade de enviar, através da rede wireless, as seguintes informações ao banco de dados localizado no computador de apoio: (i) as 24 leituras dos sensores ultra-som, (ii) as coordenadas x, y e posição angular a partir do encoder do robô, (iii) a velocidade translacional e rotacional do robô naquele momento, (iv) o rótulo numérico da ação que foi realizada pelo robô de acordo com as configurações do ambiente e (v) as quatro distâncias resumidas.

No caso da rede MLP e da Elman, são ainda acrescentados as ativações dos neurônios da camada oculta. Essas informações são coletadas à medida que o SCITOS G5 navega pela sala, realizando quatro voltas. A coleta de dados é realizada a uma taxa de 9 amostras por segundo e gerou um banco de dados com 5456 exemplos. Estes dados são então utilizados para o treinamento dos classificadores neurais apresentados anteriormente, a fim de verificar qual deles melhor "imita"o algoritmo.

A disposição dos objetos localizados na sala de teste pode ser visto na Figura 3.3.

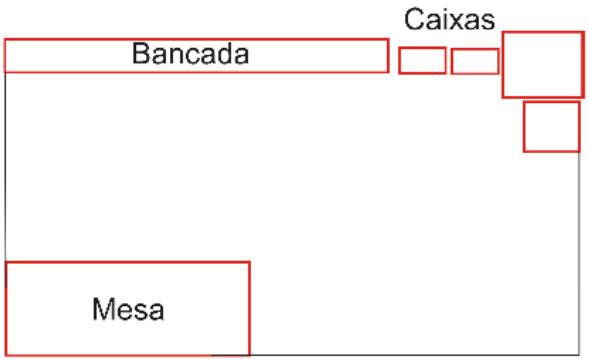

Figura 3.3: Diagrama do ambiente de navegação.

Os movimentos executados a partir da análise das distâncias resumidas à frente e à esquerda são: seguir em frente, curva aberta à direita, curva fechada à direita e curva aberta à esquerda.

Como já foi mencionado, o problema de navegação é formulado como um problema de classificação de padrões. Neste caso, as leituras sensoriais dos ultra-sons são utilizadas como entradas e as classes do problema são os quatro movimentos que podem ser realizados: seguir em frente (classe 1), curva aberta à direita (classe 2), curva fechada à direita (classe 3) e curva aberta à esquerda (classe 4).

Na Figura 3.4 é mostrada a trajetória que o robô percorreu, em pontos azuis, e a sua perspectiva do ambiente (paredes, obstáculos, etc) em pontos pretos, dada pelas distâncias retornadas pelos sensores de ultra-som.

3.2 Entradas e saídas da rede ELM

Nesta seção descrevemos as entradas e saídas da rede ELM implementada para o modelagem do robô SCITOS G5.

3.2.1 Entradas

As entradas da rede serão as leituras dos sensores do robô, o robô possui 24 sensores como mencionamos anteriormente, mas na prática vão ser usados para os treinamentos e testes 24, 4 e 2 sensores. Estas saídas vão a interatuar com a camada oculta estão representadas pelos vetores $x_0, x_1 ... x_N$.

Pode-se comprovar finalmente que ótimos resultados foram encontrados para as leituras de dois sensores como entradas da rede, utilizasse 4000 leituras de ambas velocidades para a realização do treinamento e as 1456 restantes para os testes.

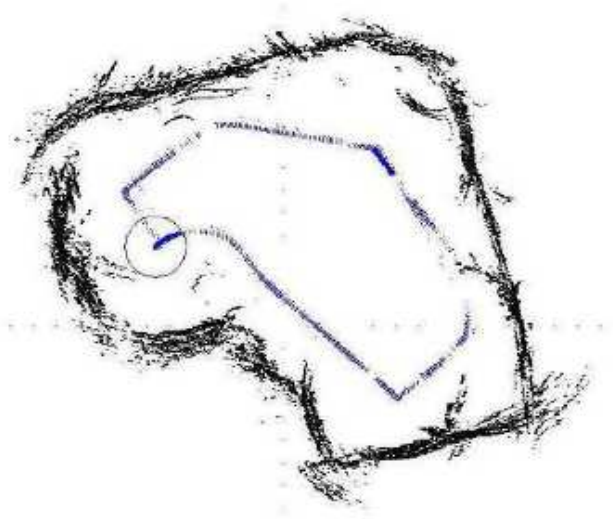

Figura 3.4: Percepção do ambiente e trajetória do SCITOS G5.

3.2.2 Saídas

As saídas da rede são duas, a velocidade translacional e a velocidade rotacional, as saídas mesmas da camada de saída depois de passar pela camada oculta. Estas variáveis são as utilizadas para os casos de estudo, são as que verificam o resultado da modelagem estática das entradas. Representadas pelos vetores $d_0, d_1 ... d_N$.

A velocidade translacional vai governar o movimento das rotas do robô pelo caminho que vai aprendendo. A velocidade rotacional vai governar o giro das rotas do robô pelo caminho aprendido. O conjunto das duas saídas é o que os neurônios vão aprender para depois poder implementar a rede neural no robô SCITOS G5 e ele tenha a capacidade de realizar de novo o caminho aprendido.

3.2.3 Aprendizagem

A taxa de aprendizagem da RNA depende de vários fatores controláveis que são ser tidos em conta. Obviamente, uma baixa taxa de formação é igual a precisamos de gastar muito tempo para realizar o treinamento produzir RNA e bem treinada. Com valores mais treinamento, a rede pode não ser capaz de discriminar tão bom quanto um sistema Saiba mais lentamente.

Geralmente, fatores adicionais além do tempo devem ser considerados quando se discute o treinamento:

- Complexidade da rede: paradigma, tamanho, arquitetura.
- Tipo de algoritmo de aprendizado utilizado.
- O nível de tolerância da rede finalmente.

Se fizermos mudanças em qualquer desses fatores, pode aumentar o tempo de treinamento ou obter um erro inaceitável.

Capítulo 4

Resultados

Os resultados dos testes realizados a partir da metodologia descrita no capítulo 3 são resumidamente apresentados. Foram realizados várias simulações, utilizando as medições de 24, 4 e 2 sensores, porém apenas os testes para dois sensores são reportados por apresentarem os melhores resultados.

4.1 Treinamento da rede ELM

A fase do treinamento da rede foi implementado no código Matlab do algoritmo ELM, o treinamento da rede foi feito com 4000 leituras dos dois sensores. Os 24 sensores estão distribuídos ao redor do robô, mas só as leituras dos sensores localizados perto da parede é que possuem alguma informação de proximidade. Na verdade, as medições feitas com os 24 sensores só adicionam ruído ao processo de aprendizado da rede ELM, uma vez que a maior parte dos sensores fica inativo durante a tarefa de navegação seguindo paredes.

4.2 Velocidade Translacional

Nesta seção mostram-se os resultados da velocidade translacional com os testes da rede ELM partindo das leituras de dois sensores para uma quantidade crescente de neurônios da camada escondida ($q = 5, 20, 50, 100, 200$).

4.2.1 Visualização do Sinal Estimado

As figuras 4.1, 4.2, 4.3, 4.4 e 4.5, ilustram a variação dos resultados mudando a quantidade de neurônios da camada escondida: quanto maior esta quantidade, melhor tende a ser o aprendizado do mapeamento de interesse. Na velocidade translacional, é visível a melhora da aproximação com o aumento da quantidade de neurônios, o que serve como indicação visual de uma boa modelagem, pelo menos em princípio. Contudo, além da análise visual, faz-se necessário testar algumas hipóteses de modelagem, tais como a de resíduos com média nula e distribuídos de forma aproximadamente gaussiana, além de resíduos com função de autocorrelação similar ao ruído branco.

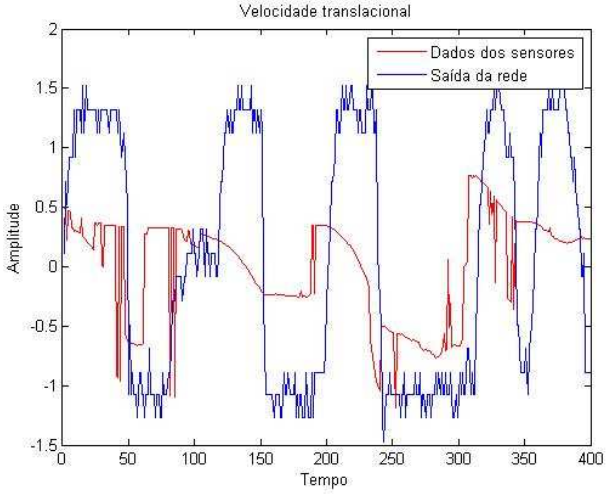

Figura 4.1: Saída velocidade translacional para 5 neurônios.

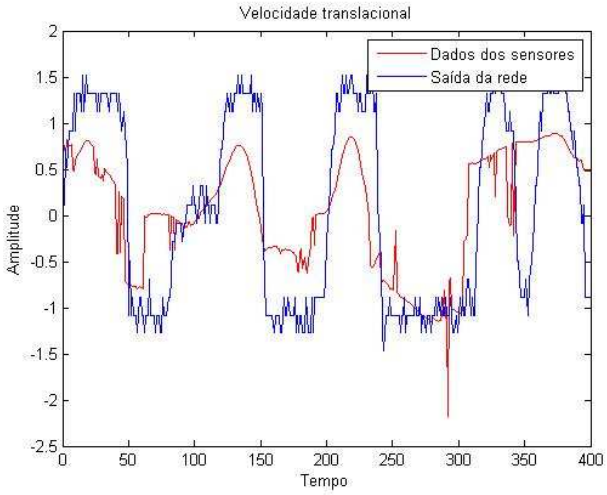

Figura 4.2: Saída velocidade translacional para 20 neurônios.

4.2.2 Histogramas dos Resíduos (Erro de Estimação)

Nas seguintes figuras 4.6, 4.7, 4.8, 4.9 e 4.10, pode-se verificar os histogramas dos erros para os diferentes testes. A distribuição dos erros tende a ser muito parecida, com os histogramas dos erros assumindo progressivamente uma forma mais similar da distribuição normal à medida que o número de neurônios aumenta. Além disso, cada vez mais, a concentração dos valores dos resíduos quase fica só entorno do valor zero, o que quer dizer que o erro é aceitável para fazer o modelagem da velocidade translacional.

4.2.3 Função de autocorrelação dos resíduos

As autocorrelações são calculadas para os sinais de erros (resíduos) obtidos nos testes. Nas figuras 4.11, 4.12, 4.13, 4.14 e 4.15, pode-se observar que os

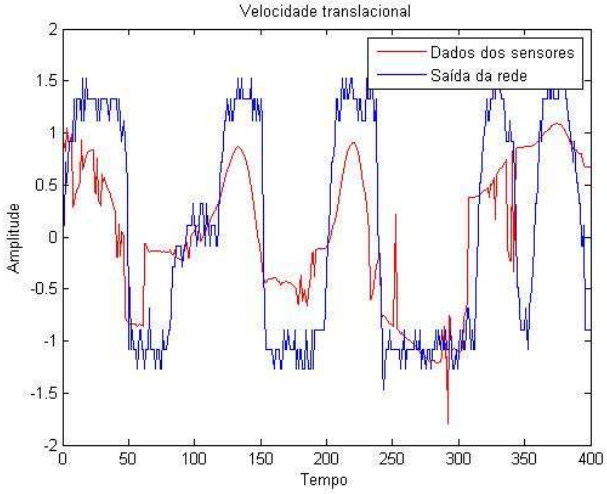

Figura 4.3: Saída velocidade translacional para 50 neurônios.

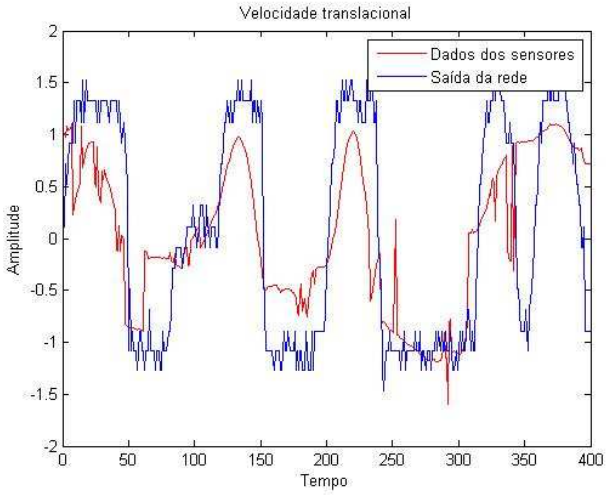

Figura 4.4: Saída velocidade translacional para 100 neurônios.

valores da autocorrelação são altos, mantendo o padrão mesmo com o aumento do número de neurônios escondidos. O ideal é os valores da autocorrelação fossem todos nulos, exceto para o atraso (lag) zero. Isto é indicativo da presença de dinâmica não-modelada, ou seja, as entradas no instante atual (n) não são suficientes para capturar toda a dinâmica da tarefa contida nos dados. Para isso, seria preciso introduzir algum mecanismo de memória no processo de aprendizagem. Isto pode ser feito introduzindo-se tanto os valores atuais das leituras dos sensores, quanto valores passados $(n-1, n-2, ...)$.

4.2.4 Resumo dos resultados dos erros da velocidade rotacional

Na Tabela 4.1 pode se observar os valores do erro quadrático médio obtidos para a tarefa de modelagem da velocidade translacional, variando-se o número de neurônios da camada escondida. Percebe-se que à medida que usamos uma

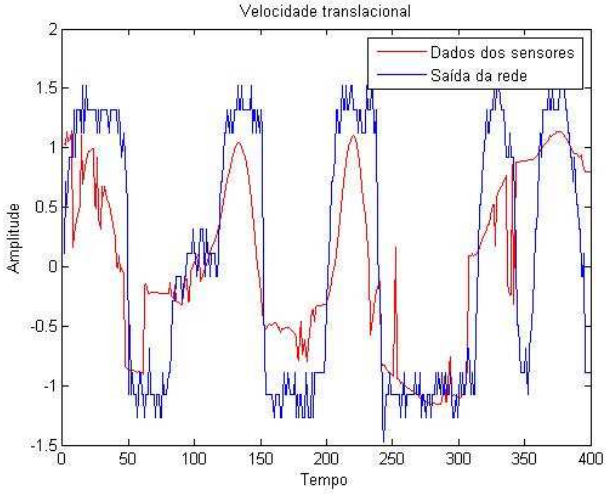

Figura 4.5: Saída velocidade translacional para 200 neurônios.

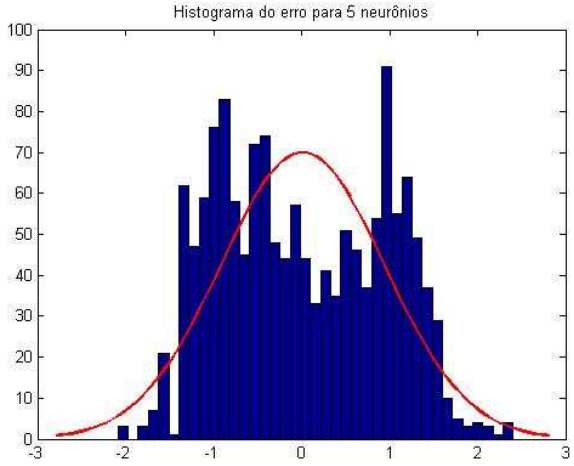

Figura 4.6: Histograma do erro para 5 neurônios.

quantidade maior de neurônios, o erro decresce.

4.3 Velocidade Rotacional

Nesta seção são mostrados os resultados obtidos para a velocidade rotacional com os testes da rede ELM, utilizando as leituras de dois sensores e para uma quantidade variável neurônios na camada escondida, ou seja,$q = \{5, 20, 50, 100, 200\}$.

4.3.1 Visualização do sinal estimado

Nas seguintes figuras 4.16, 4.17, 4.18, 4.19 e 4.20, mostra-se a variação dos resultados mudando os neurônios utilizados na camada escondida. Ao contrário do que ocorreu para a velocidade translacional, a rede ELM não é

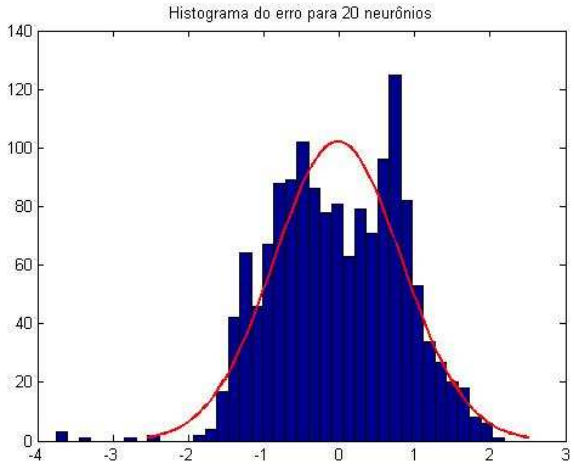

Figura 4.7: Histograma do erro para 20 neurônios.

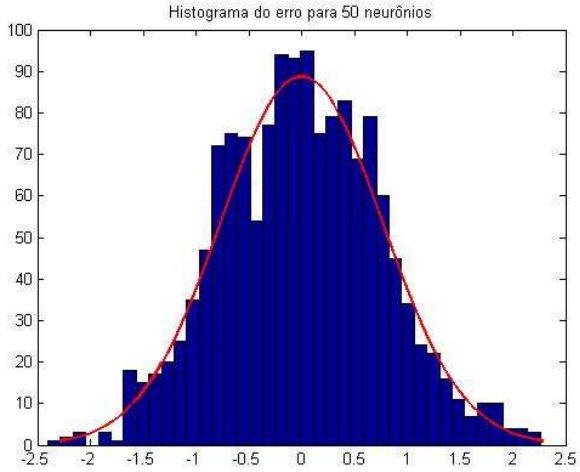

Figura 4.8: Histograma do erro para 50 neurônios.

capaz de aprender o mapeamento entrada-saída de interesse (leituras sensoriais $\mapsto$ velocidade rotacional), mesmo variando-se o número de neurônios na camada escondida.

Uma possível explicação para isso está no fato de a velocidade rotacional apresentar elevada variação (freqüência), bem maior do que a apresentada pela velocidade translacional. Esta maior freqüência pode ser o reflexo de uma dinâmica bem mais complexa, sendo necessário a incorporação de mecanismos de memória para modelar adequadamente o fenômeno.

4.3.2 Histograma dos Resíduos (Erros de Estimação)

Nas seguintes figuras 4.21, 4.22, 4.23, 4.24 e 4.25, pode-se verificar os histogramas dos erros para os diferentes testes efetuados. Para todos as quantidades de neurônios ocultos testados, os histogramas estão longe de

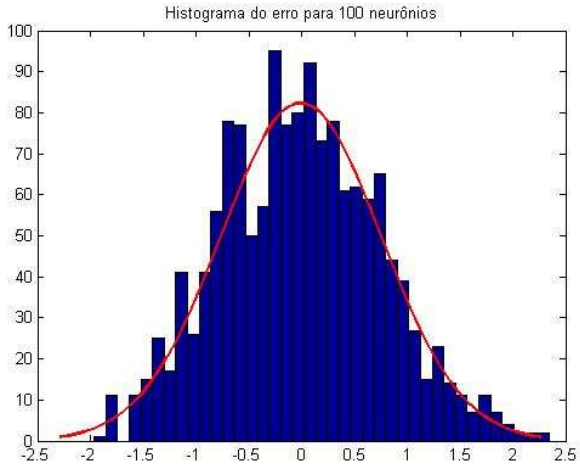

Figura 4.9: Histograma do erro para 100 neurônios.

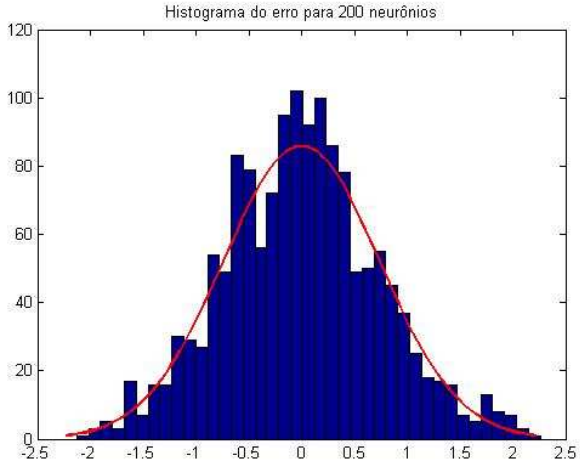

Figura 4.10: Histograma do erro para 200 neurônios.

assumir formas similares a uma distribuição gaussiana, fato indicativo de uma modelagem pobre do fenômeno. Poderia ser melhorado como antes foi mencionado, com a introdução de memória na modelagem.

4.3.3 Função de Autocorrelação dos Resíduos

As funções de autocorrelação foram calculadas para os erros de estimação obtidos nos testes. Nas figuras **??**, 4.27, **??**, 4.29 e **??**, pode se observar que os erros de estimação são baixos, o que em princípio seria indicativo de uma boa modelagem, a tendência da função de autocorrelação é ótima aproximando se ao zero. Contudo, quando este resultado é conjugado ao dos histogramas dos resíduos e ao da avaliação visual, percebe-se que a modelagem é ruim como um todo.

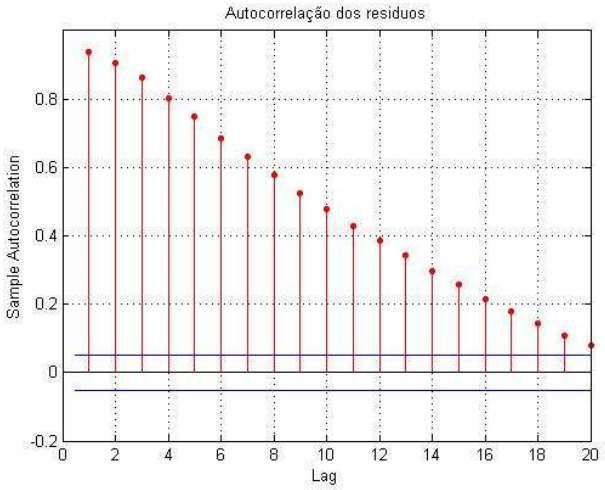

Figura 4.11: Autocorrelação para 5 neurônios.

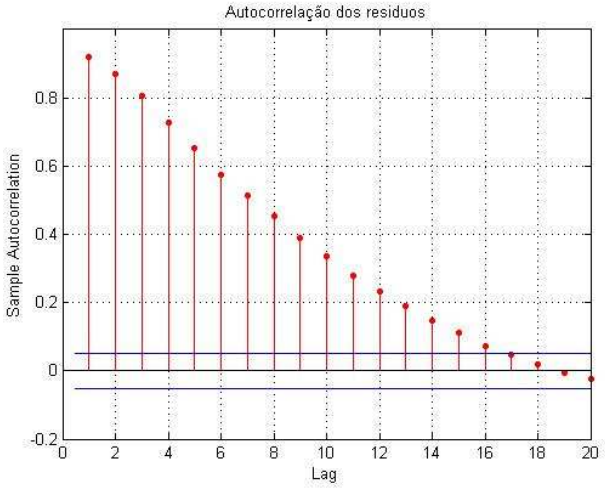

Figura 4.12: Autocorrelação para 20 neurônios.

4.3.4 Resumo dos resultados dos erros da velocidade rotacional

Na Tabela 4.2 pode se observar os erros da velocidade rotacional, obtidos nos testes da rede, mudando o número de neurônios da camada escondida. Embora, haja uma ligeira diminuição do erro quadrático médio com o aumento do número de neurônios, esta diminuição é desprezível (muito pequena), de modo que pode-se afirmar que a melhoria no desempenho da rede é insignificante com o aumento do número de neurônios ocultos.

Tabela 4.2:

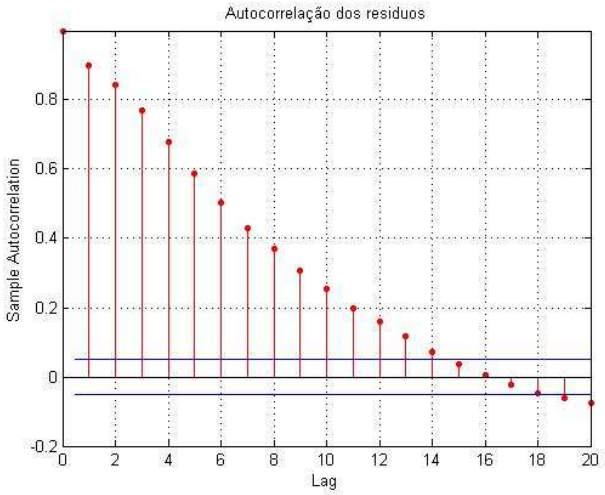

Figura 4.13: Autocorrelação para 50 neurônios.

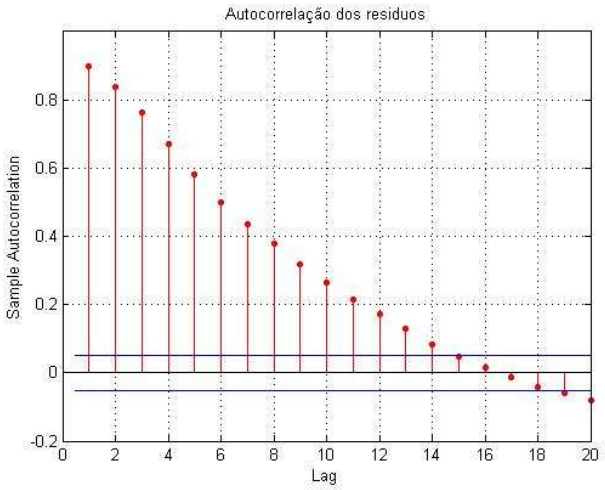

Figura 4.14: Autocorrelação para 100 neurônios.

Tabela 4.1: Erros da velocidade translacional para diferentes números de neurônios.

Neurônios hline 5	EQM 0,8685	EQM Normalizado 6,048
20	0,7123	2,6232
50	0,581	1,5022
100	0,5576	1,2871
200	0,564	1,2556

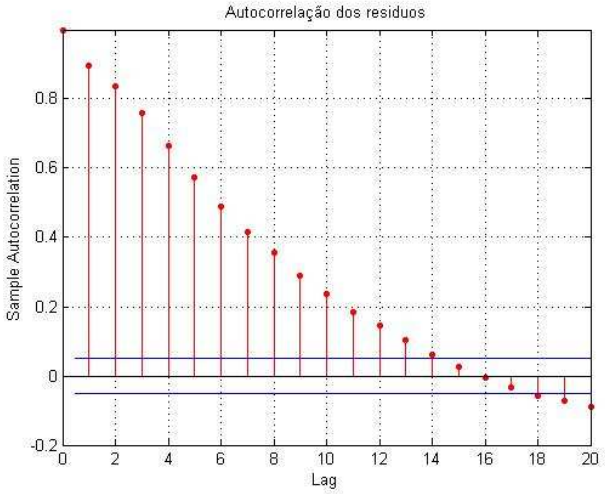

Figura 4.15: Autocorrelação para 200 neurônios.

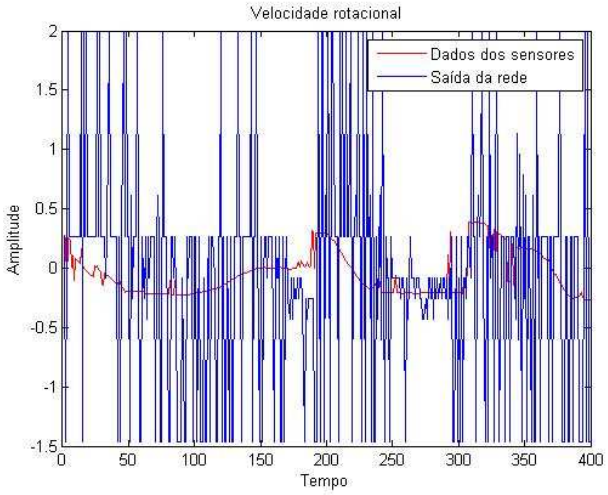

Figura 4.16: Saída velocidade rotacional para 5 neurônios.

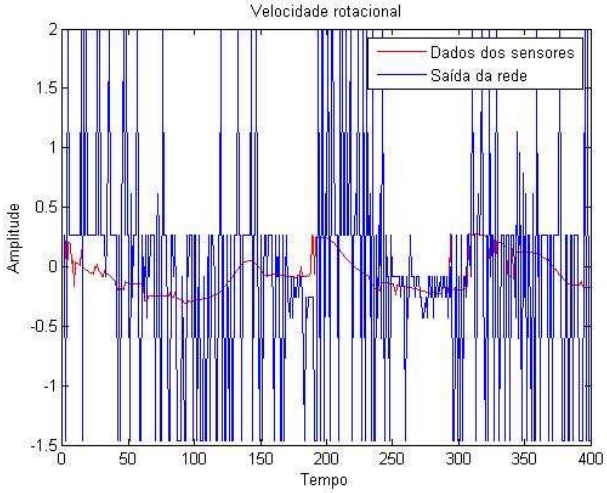

Figura 4.17: Saída velocidade rotacional para 20 neurônios.

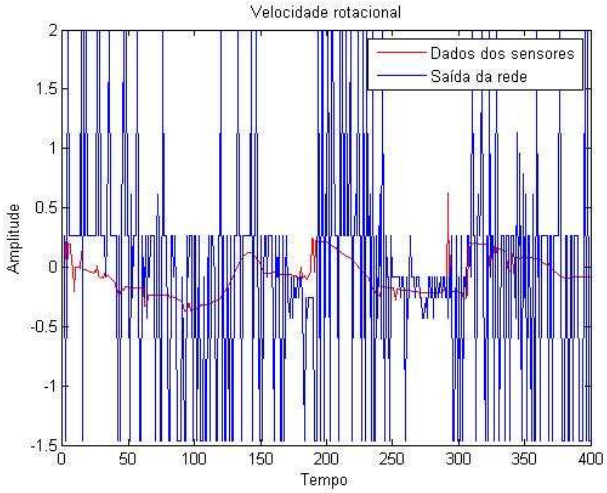

Figura 4.18: Saída velocidade rotacional para 50 neurônios.

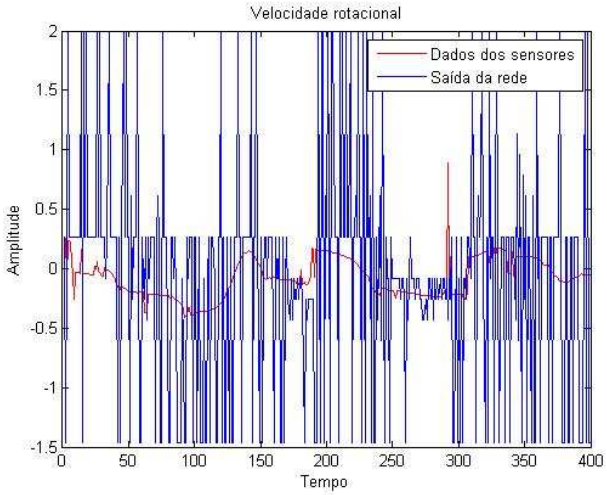

Figura 4.19: Saída velocidade rotacional para 100 neurônios.

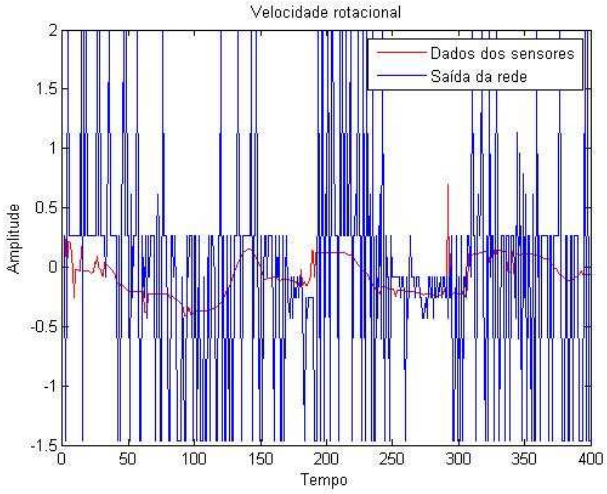

Figura 4.20: Saída velocidade rotacional para 200 neurônios.

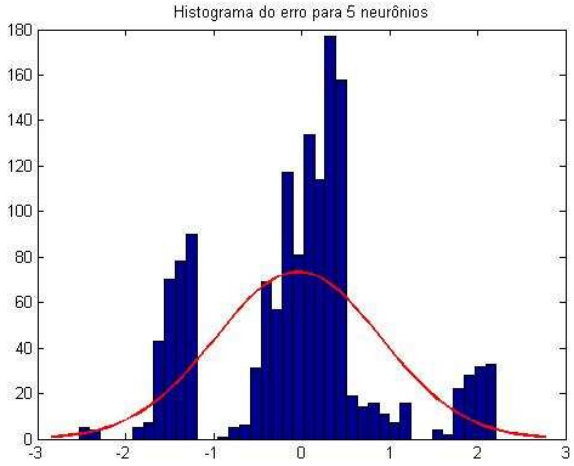

Figura 4.21: Histograma do erro para 5 neurônios.

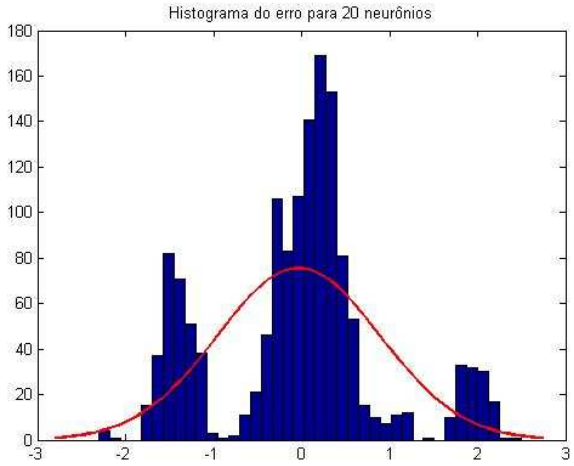

Figura 4.22: Histograma do erro para 20 neurônios.

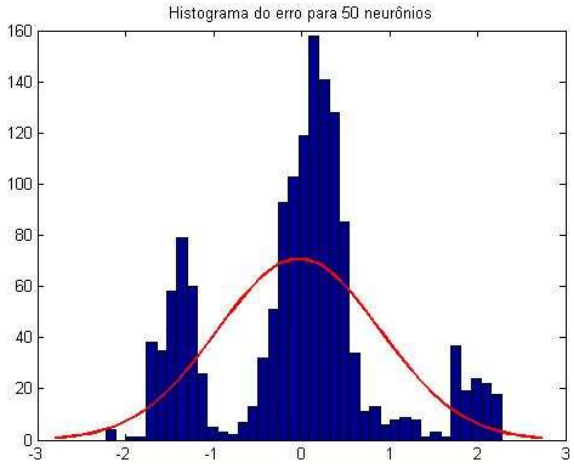

Figura 4.23: Histograma do erro para 50 neurônios.

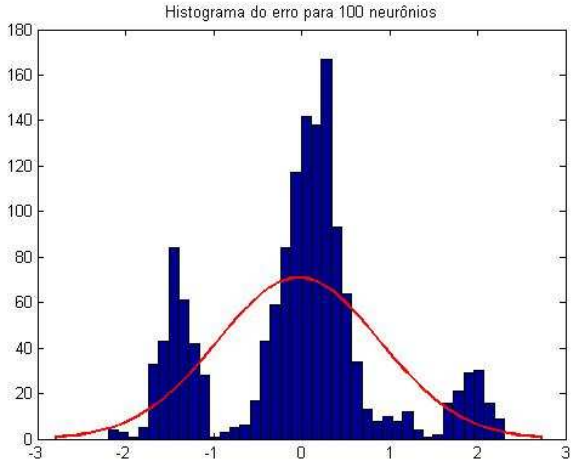

Figura 4.24: Histograma do erro para 100 neurônios.

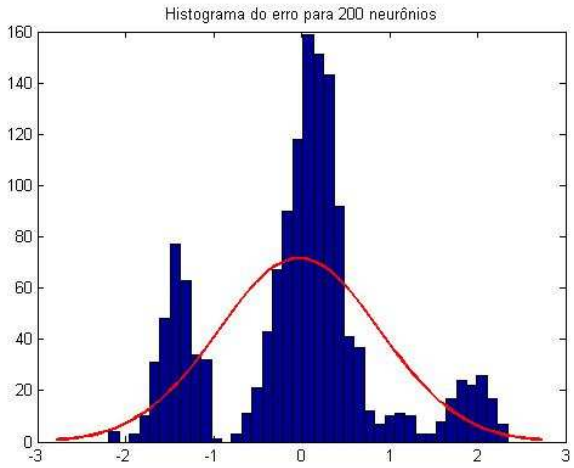

Figura 4.25: Histograma do erro para 200 neurônios.

Capítulo 5

Conclusões e Trabalhos Futuros

Neste capítulo são feitas considerações finais do trabalho realizado. A seção 5.1 vai descrever as conclusões principais derivadas do estudo dos resultados da modelagem estática, a seção 5.2 descreve as limitações surgidas no desenvolvimento do projeto e a seção 5.3 aborda os possíveis perspectivas, proportas e trabalhos futuros.

5.1 Conclusões

Verificamos que a aprendizagem é possível na simulação, a rede neural é capaz de imitar ao cérebro humano, conceito muito longe dos meios físicos que temos a disposição para estudo, mas obrigados à implementação da rede ELM pode se comprovar.

Do modelo visual podemos concluir que a modelagem da velocidade translacional pode ser aceitável porque se acerca ao modelo que queremos conseguir, mas nas suas funções de autocorrelação se observa que distam muito da autocorrelação ótima que seria de uma distribuição normal e teria valores menores. O contrario acontece com a velocidade rotacional, a qual tem uma modelagem não aceitável na saída dos resultados visuais, mas tem uma função de autocorrelação dos resíduos ótima.

O ideal seria ter esses dois estudos pelo menos aceitáveis para poder prosseguir ao seguinte passo, a modelagem dinâmica do mapeamento das leituras sensoriais de ambas velocidades.

Os testes forem feitos usando leituras dos dois sensores ultra-som do robô que não introduzem ruído ao sistema, podem ser utilizadas para poder introduzir informação temporal(dinâmica) no processo de modelagem do robô e pode pegar os valores atuais e passados de cada sensor. Esse fato pode melhorar os resultados.

5.2 Limitações

Na hora de fazer os testes da velocidade rotacional, a rede não consegue fazer uma boa aprendizagem devido aos valores da velocidade rotacional, esses são muito abruptos e apresenta elevada variação o que diz dela que tem uma dinâmica mais complexa que a da velocidade translacional.

Outra observação foi nos testes de 24 e 4 sensores, no modelo visual das velocidades rotacional e translacional, os resultados apresentavam-se com índice maior de erros, obtendo uma modelagem aceitável mas sem descrever de boa forma o comportamento das entradas do sistema (leituras sensoriais das duas velocidades).

5.3 Perspectivas Futuras

Como foi mencionado na seção 5.1, a obtenção dos dados, as leituras sensoriais podem se utilizar para introduzir memória ao robô, assim, este pode aprender baseando se nos neurônios da rede. Com isso poderíamos realizar a modelagem dinâmica do mapeamento das leituras sensoriais com o fim de utilizar essa memória adquirida por a modelagem estática do mapeamento das leituras. Introduzir memória num robô móvel pode ser um bom passo na união das duas matérias, a ciência e a computação.

Como proposta, depois de fazer o estudo dos resultados dessas duas modelagens, o seguinte passo seria implementar no robô SCITOS G5 a memória, e obter sucesso na sua aprendizagem.

No campo da Ciência da Computação, Inteligência Artificial é uma das áreas de maior expectativa, até mesmo na sociedade em geral, pois a busca para compreender os mecanismos de inteligência, foi um Santo Graal da obra de muitos cientistas por muitos anos e ainda é. Dentro da área de inteligência artificial que mais tem atraído a aprendizagem de máquina é vital para emular o processo de comportamentos inteligentes. Esse sistema pode melhorar o seu desempenho com base na experiência recolhida para executar uma tarefa repetitiva e também tem uma noção do que é errado e você pode evitá-lo, é emocionante.

No mercado de robótica em que há também a inteligência artificial, robôs e máquinas, a tecnologia tem um grande futuro e é para o futuro, e espera que esse mercado vai se expandir para além das grandes empresas. Ao implantar aos robôs de muitos mais atributos dos seres humanos. É bastante visível a grande aplicação da robótica em todos os campos industriais. Mas outra coisa que acontece é que devido ao alto custo que representa o processo de produção automatizada e robotizada, a tendência atual na área de robótica é a investigação de micro-robôs e robôs com um grau de inteligência para atingir esse processo de produção.

Referências Bibliográficas

AHALT S. C.; KRISHNAMURTHY, A. K. C. P. M. D. E. Neural networks. *IEEE Journal on Selected Areas in Communication*, v. 3, n. , p. 277–290, . .

BARRETO, P. D. G. de A. *EXTREME LEARNING MACHINE (ELM)*. Dissertação — Universidade Federal do Ceará(UFC), Fortaleza-CE, Junho 2010. Programa de Pós-Graduação em Engenharia de Teleinformatica (PPGETI),Departmento de Engenharia de Teleinformatica.

KRISHNA, K. e. P. K. K. Solving the local minima problem for a mobile robot by classification of spatio-temporal sensory sequences. *Journal of Robotic System*, , n. , p. 549–564, September 2000. .

KUNG, S. Y. *DIGITAL NEURAL NETWORKS*. [S.l.: s.n.], 1993.

MCCLELLAND, D. R. e J. *Parallel Distributed Processing (Processamento Distribuído Paralelo)*. [S.l.: s.n.], 1986.

ROSEMBLATT. *Principles of Neurodynamics*. [S.l.: s.n.], 1958.