UNIVERSIDADE DO VALE DO RIO DOS SINOS
CIÊNCIAS EXATAS E TECNOLÓGICAS
PROGRAMA INTERDISCIPLINAR DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO
APLICADA - PIPCA

Extração de Conhecimento a partir de Redes Neurais Recorrentes

por

DENISE REGINA PECHMANN SIMON

Dissertação submetida à avaliação
como requisito parcial para a obtenção do grau de
Mestre em Computação Aplicada

Prof. Dr. Adelmo Luis Cechin
Orientador

São Leopoldo, maio de 2004.

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Simon, Denise Regina Pechmann

Extração de Conhecimento a partir de Redes Neurais Recorrentes / por Denise Regina Pechmann Simon. — São Leopoldo: Ciências Exatas e Tecnológicas da UNISINOS, 2004.

113 f.: il.

Dissertação (mestrado) — Universidade do Vale do Rio dos Sinos. Ciências Exatas e TecnológicasPrograma Interdisciplinar de Pós-Graduação em Computação Aplicada - PIPCA, São Leopoldo, BR–RS, 2004. Orientador: Cechin, Adelmo Luis.

1. Redes Neurais Recorrentes. 2. Máquina de Estados Finita. 3. Lógica Difusa. I. Cechin, Adelmo Luis. II. Título.

UNIVERSIDADE DO VALE DO RIO DOS SINOS
Reitor: Dr. Aloysio Bohnen
Pró-Reitor Acadêmico: Padre Dr. Pedro Gilberto Gomes
Diretora de Programas de Pós-Graduação e Pesquisa: Profª. Drª. Ione Bentz
Coordenador do PIPCA: Prof. Dr. Arthur Tórgo Gómez

"Ao meu Pai,
por sempre ter acreditado em mim! "

Agradecimentos

Em primeiro lugar gostaria de agradecer ao meu orientador Doutor Adelmo Luis Cechin pelo seu apoio e incentivo. De fato, durante a elaboração deste trabalho esteve sempre disponível para analisá-lo e sugerir possíveis soluções para os problemas encontrados. Por outro lado, sempre me proporcionou o espaço necessário para seguir à minha maneira os caminhos que foram definidos. Por tudo isso, o meu obrigada.

Gostaria também de agradecer ao Prof. Doutor Luiz Paulo Luna de Oliveira pela sua disponibilidade para analisar, comentar e sugerir melhoramentos ao trabalho realizado.

Reconheço e agradeço também o suporte financeiro e material disponibilizado pela CAPES e pela UNISINOS.

Aos funcionários do PIPCA, pela atenção e ajuda prestada durante estes dois anos de trabalho, assim como aos professores que através de suas palavras me fizeram prosseguir e chegar até aqui.

Agradeço, ainda, aos meus colegas e amigos, por estarem ao meu lado nos momentos mais difíceis.

Um agradecimento especial à minha família, principalmente à minha Mãe, pelo amor, amizade, carinho e por sempre estarem comigo quando precisei.

Finalmente, ao Alexandre por ter me ajudado para que a realização deste mestrado fosse possível (e ao Nichaw e à Mui Key por me acompanharem durante as madrugadas de trabalho).

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

RNA	Rede Neural Artificial
MLP	Multilayer Perceptron
AF	Autômatos Finitos
AFDet	Autômato Finito Determinístico
AFND	Autômato Finito Não Determinístico
AFDif	Autômato Finito Difuso
HMM	Hidden Markov Model
TDNN	Time Delay Neural Network
RNR	Rede Neural Recorrente
NARX	Nonlinear Autoregressive with Exogenous Inputs
RMLP	Recurrent Multilayer Perceptron
BPTT	Back-Propagation Through Time
SNNS	Stuttgart Neural Network Simulator
FSA	Finite State Automata Utilities

TITLE: "KNOWLEDGE EXTRACTION FROM RECURRENT NEURAL NETWORKS"

Abstract

In this work a method of knowledge extraction from Recurrent Neural Network is proposed. Express formally the knowledge stored inside an Artificial Neural Network is a great challenge, because such knowledge has to be reformulated and presented by simple and understandable means. Three symbolic formats are presented for the representation of this knowledge: Fuzzy Finite Automata, Markov Chains and Deterministic Finite Automata. For the knowledge extraction used in this work, each space region of the neuron activity is associated to a meaning. The considered method uses clusterization of the neural space in order to obtain the automata states, using the K-means algorithm and the fuzzy clustering. The knowledge acquisition is made using Recurrent Neural Networks to learn the behavior of the two non linear dynamic systems and, from the trained nets, to extract the states and possible automata transitions. The dynamic systems are the Inverse Pendulum system and the Lorenz system. The presented extraction method was validated with the knowledge representation of the two studied dynamic systems, through the three symbolic formats.

Keywords: Recurrent Neural Network, Finite State Machine, Fuzzy Logic.

Resumo

Neste trabalho é proposto um método de extração de conhecimento a partir de Redes Neurais Recorrentes. Expressar formalmente o conhecimento armazenado dentro de uma Rede Neural Artificial representa um grande desafio, já que tal conhecimento precisa ser reformulado e apresentado de uma maneira simples e inteligível. Três formalismos simbólicos são abordados para a representação deste conhecimento: Autômatos Finitos Difusos, Cadeias de Markov e Autômatos Finitos Determinísticos. Para as extrações de conhecimento utilizadas no trabalho, atribui-se significado às regiões do espaço de atividade dos neurônios. O método proposto utiliza a clusterização do espaço neural para obtenção dos estados do autômato, sendo utilizados para isso, o algoritmo *K-means* e a clusterização difusa. A obtenção do conhecimento é feita utilizando-se Redes Neurais Recorrentes para aprender o comportamento de dois sistemas dinâmicos não lineares e, a partir das redes treinadas, extrair os estados e possíveis transições do autômato. Os sistemas dinâmicos utilizados para ilustrar este tipo de extração de conhecimento, são o sistema do Pêndulo Inverso e o sistema de Lorenz. O método de extração apresentado foi validado com a representação do conhecimento dos dois sistema dinâmicos estudados, através dos três formatos simbólicos.

Palavras-chave: Redes Neurais Recorrentes, Máquina de Estados Finita, Lógica Difusa.

Capítulo 1

Introdução

Assim como é difícil para muitos especialistas expressar formalmente o seu conhecimento possibilitando a utilização em um programa de computador, extrair o conhecimento de modelos baseados no funcionamento do cérebro, como as Redes Neurais Artificiais, representa um grande desafio.

A principal razão para o estudo sobre extração de conhecimento está na dificuldade apresentada pelos seres humanos de interpretar a representação numérica de uma Rede Neural. O conhecimento pode ser formalmente obtido, ou pode ser representado através de um sistema que utiliza Redes Neurais, em uma forma humanamente compreensível. Por humanamente compreensível entende-se que o conhecimento armazenado em um sistema neural é também representável em uma forma simbólica, ou uma forma bem estruturada, como funções Booleanas, autômato, regras, ou em um modelo similar de boa compreensão.

Entender o conhecimento armazenado dentro de um sistema pode tornar os sistemas neurais mais aceitáveis em um grande número de áreas de aplicação. Sistemas que utilizam Redes Neurais tem demonstrado que são capazes de modelar muito bem alguns aspectos da inteligência humana, como a visão e a fala [13]. Uma característica particularmente atraente é sua habilidade para adquirir conhecimento pela aprendizagem de dados. O problema é que a representação interna de conhecimento desenvolvida pelas Redes Neurais é amplamente incompreensível para humanos, e assim não pode ser manipulada facilmente.

Redes Neurais Artificiais são conhecidas pelo bom desempenho que geralmente obtêm quando utilizadas em um grande variedade de aplicações. No entanto, para muitas aplicações, não é importante apenas o desempenho obtido, mas também a facilidade de o usuário compreender como a rede chega às suas decisões.

Uma das razões porque sistemas especialistas são mais facilmente aceitos que Redes Neurais, é sua capacidade para explicar como chegaram à uma solução de um determinado problema. A explicação para isso, é o uso de um processo de raciocínio automatizado que usa a base de conhecimento e um conjunto de regras descrevendo um domínio. Redes Neurais, de outra forma, não fornecem uma explicação como parte de seu processamento de informação. O conhecimento que a Rede Neural ganha através do treinamento é armazenado em seus pesos. Até recentemente, foi extensamente aceito o mito que Redes Neurais são "caixas pretas", ou seja, o conhecimento

armazenado em seus pesos após o treinamento não era acessível por inspeção, análise e verificação. Desde então, pesquisas neste tópico tem resultado em um número de algoritmos para extração de conhecimento em forma simbólica a partir de Redes Neurais Artificiais treinadas [13].

Cada vez mais, acredita-se que todo potencial das Redes Neurais não poderá ser completamente explorado enquanto não for acrescentado a estes modelos um mecanismo que explique e justifique suas decisões.

A aquisição de conhecimento por uma Rede Neural é feita através de um processo automático de aprendizado e os seus pesos sinápticos são usados para armazenar o conhecimento. Estas características fazem das Redes Neurais uma ferramenta fundamental para automatizar o processo de aquisição do conhecimento. Entretanto, devido à sua natureza distribuída, expressar este conhecimento de uma forma simples torna-se uma tarefa difícil. Atualmente, o problema genérico da extração de conhecimento a partir de redes neurais é tratado pelos sistemas híbridos. Estes são sistemas computacionais baseados principalmente em Redes Neurais, mas que permitem também uma interpretação simbólica de seus componentes ou que interagem com componentes simbólicos [86]. Alguns desses sistemas podem ser encontrados em [10], [13], [4] e [22].

Para redes *feedforward* o conhecimento tem sido extraído tipicamente em forma de cláusulas/orações *if-them* Booleanas e difusas. Para Redes Recorrentes, Autômatos Finitos tem sido o principal paradigma de extração de conhecimento simbólico temporal [60], [29] e [27]. Claramente Redes Neurais não são mais "caixas pretas". Algumas aplicações (como, aplicações de Redes Neurais para avaliação de crédito e política de empréstimos e aplicações críticas como controle aéreo) podem requer que Redes Neurais sofram validação antes de serem inicialmente aplicadas. A extração de conhecimento pode ser um importante estágio para este processo.

Inúmeras aplicações de Redes Neurais Temporais podem ser encontradas em importantes problemas atuais como na previsão e modelagem de séries temporais, no processamento de sons (voz) e imagens, na previsão de indicadores financeiros, controle de processos, simulação numérica e visualização, sequenciamento de DNA, proteínas, etc.. Entretanto, para várias aplicações, é importante não apenas a obtenção do conhecimento, mas também a facilidade do usuário compreender como e por que determinadas decisões estão sendo tomadas pela Rede Neural. A extração do conhecimento provê uma interface entre o profissional na área da aplicação e o conhecimento contido nos modelos de Redes Neurais.

Através da representação em um formalismo simbólico o profissional pode então, validar o modelo neural baseado em seu conhecimento prévio sobre o processo antes de admiti-lo como adequado, repensar ou alterar a estratégia neural, os métodos de medição, os sensores utilizados, com o objetivo de encontrar melhores soluções para um problema, trazendo benefícios em termos de credibilidade para estes profissionais e para um uso mais adequado das Redes Neurais.

Redes Neurais Recorrentes são usadas na modelagem de séries temporais de processos industriais, indicadores financeiros, simulações em computadores, séries geradas por sistemas dinâmicos, entre outras, sendo consideradas sistemas dinâmicos não lineares. Neste trabalho investiga-se o problema da extração de conhecimento armazenado em Redes Neurais Recorrentes, treinadas com séries temporais, sob a forma de Autômatos Finitos Determinísticos, Autômatos Finitos Difusos e Cadeias de Markov.

O Capítulo 2 apresenta os conceitos básicos, iniciando pela seção sobre a Rede Neural do tipo Multilayer Perceptron. As seções seguintes apresentam alguns conceitos importantes sobre Sistemas Dinâmicos, Autômatos Finitos Determinísticos, Cadeias de Markov, Lógica e Conjuntos Difusos, que tornam-se fundamentais para a definição dos métodos de extração de conhecimento a partir de Redes Recorrentes. As Redes Neurais Temporais são apresentadas e classificadas no Capítulo 3, juntamente com uma breve discussão sobre algoritmos de aprendizagem e aplicações. Ainda no Capítulo 3, são abordados alguns dos trabalhos mais importantes relacionados ao uso de Redes Temporais, tanto para extração de conhecimento, como para síntese de conhecimento e formas para sua aplicação. O Capítulo 4 descreve as metodologias aplicadas para a extração de conhecimento a partir de Redes Neurais Recorrentes. Três tipos formalismos simbólicos foram obtidos através dos métodos de extração: Autômatos Finitos Determinísticos, Cadeias de Markov e Autômatos Finitos Difusos. As duas aplicações utilizadas para a extração de conhecimento, o Sistema do Pêndulo Inverso com controlador e o Sistema de Lorenz, são descritas no Capítulo 5. Os resultados obtidos para cada uma das aplicações, são apresentados e descritos no Capítulo 5, juntamente com suas respectivas aplicações. Finalizando, apresentam-se as conclusões obtidas através dos estudos e dos experimentos executados, e algumas idéias interessantes para trabalhos futuros.

1.1 O Problema

O problema estudado é a extração de conhecimento formal a partir de Redes Neurais Temporais, mais precisamente de Redes Neurais Recorrentes. Como encontrar um modelo formal que propicie a visualização e o entendimento sobre o conhecimento representado dentro de uma Rede Neural Recorrente.

Em [20], [81], [61] demostrou-se que, a partir de Redes Neurais Recorrentes treinadas com seqüências temporais simbólicas, um Autômato Finito Determinístico completo e suas classes equivalentes podem ser extraídos. As ativações dos neurônios nas camadas escondidas representam o passado e os *clusters* formados a partir de grupos destas ativações representam os estados do autômato gerado.

Uma dificuldade na interpretação das Redes Neurais como Autômato Determinístico está na discretização do espaço de estados em um número finito de regiões. Redes Neurais Recorrentes, porém, podem ser sistemas dinâmicos não lineares sensíveis às condições iniciais e à dinâmica da transição de estados. Não importa o quão fina seja a discretização, eventualmente será encontrado um estado que se divide em múltiplas trajetórias resultando em uma transição de estados prevista pelo autômato [61].

Outra dificuldade reside na enorme flexibilidade da estrutura das Redes Temporais, número de entradas e número de saídas. O procedimento de modelagem de processos temporais através de Redes Neurais pode ser bastante tedioso pelo grande número de possibilidades a serem experimentadas, tendo como único parâmetro de comparação o erro em relação à um conjunto de dados de treinamento.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo deste trabalho consiste em estabelecer uma interface entre o formalismo simbólico de representação do conhecimento (simples de entender para um ser humano) e o conhecimento armazenado em Redes Neurais Recorrentes, visto que, o entendimento e visualização do conhecimento contido nas redes não se apresenta de forma clara. Fazer um estudo sobre a extração de conhecimento a partir de Redes Neurais Recorrentes, mais precisamente sobre a extração do conhecimento em forma de Autômato Finito Determinístico, Autômato Finito Difuso e Cadeias de Markov.

1.2.2 Objetivos Específicos

Entre os objetivos específicos deste trabalho estão:

- Validar a estrutura e os conteúdos aprendidos por Redes Neurais Recorrentes;
- Estudar a extração de conhecimento a partir de Redes Neurais Recorrentes suas etapas e características;
- Comparar métodos de representação do conhecimento, como: Autômatos Finitos Determinísticos, Autômatos Finitos Difusos e Cadeias de Markov;
- Comparar métodos de clusterização que possibilitam a extração de estados;
- Estudar métodos adequados para realizar a conversão de estados difusos em estados clássicos do autômato finito sem perda considerável de informação. Entre as possíveis abordagens serão consideradas, o estado como tendo um raio constante, o estado como tendo um raio variável e a utilização de diagramas de Voronoi.

Capítulo 2

Conceitos Básicos

2.1 Rede "Multi-Layer Perceptron"

Esta seção apresentada conceitos importantes sobre as redes Perceptron de Múltiplas Camadas (*Multilayer Perceptron*, MLP), que possuem características básicas fundamentais para o estudo das Redes Neurais Temporais.

As Redes Neurais Artificiais (RNAs) podem ser divididas em redes alimentadas adiante, redes alimentadas adiante com atraso no tempo e redes recorrentes. Estas duas últimas, também chamadas de redes temporais, estão baseadas na primeira, porém apresentam um comportamento mais complexo. Um exemplo de arquitetura alimentada adiante é a MLP. Este tipo de RNA apresenta pelo menos uma camada intermediária ou oculta [70].

No cérebro humano, centenas ou milhares de neurônios processam informações sobre um mesmo assunto ao mesmo tempo e enviam os resultados para os neurônios seguintes nas sinapses (conexões), gerando assim uma análise mais completa e detalhada sobre o assunto examinado do que cada neurônio teria condições de executar isoladamente. O neurônio artificial é um processador simples, ele obtém as informações do exterior ou de outros neurônios, toma uma única decisão e, por meio de um canal de saída, passa o resultado para o neurônio seguinte. Quando vários desses neurônios são ligados em rede, o efeito conjunto é a capacidade de tomar decisões complexas. Apesar das dificuldades no treinamento de RNAs e do fato de elas poderem aprender conceitos errados, esta tecnologia tem a grande vantagem de aprender com as novas situações com que se deparam, podendo se adaptar após um período de treinamento.

O aprendizado de uma RNA consiste na adaptação de seus parâmetros por um processo de estímulo contínuo do ambiente em que se encontra. O tipo de aprendizado é determinado pela forma como estes parâmetros são atualizados. A qualidade do aprendizado de uma Nede Neural pode ser medida pela diferença no desempenho que esta obtém, quando aplicada na solução de um problema antes e depois do seu treinamento. Um modo de fazer a validação da Rede Neural é utilizando uma base de dados para a fase de treinamento e outra base de dados para a fase de teste, considerando que as duas bases devem conter dados representativos do problema analisado [34].

2.1.1 Modelo MLP

As redes de uma só camada resolvem apenas problemas linearmente separáveis. A solução de problemas não linearmente separáveis passa pelo uso de redes com uma ou mais camadas ocultas.

Redes do tipo MLP utilizam como elemento básico de processamento o Perceptron, porém a sua arquitetura é definida em várias camadas, sendo a primeira, a camada de entrada e a última, a camada de saída, e entre estas duas camadas podem existir uma ou mais camadas ocultas, como pode ser visto na Figura 2.1. Os neurônios ocultos capacitam a rede a aprender tarefas complexas extraindo as características mais significativas dos padrões de entrada.

A camada de entrada da rede propaga um sinal de entrada para a primeira camada oculta, sendo que as saídas resultantes desta camada, são por sua vez aplicadas à próxima camada oculta, e assim por diante para o resto da rede. Os neurônios ocultos e de saída de uma MLP são projetados para realizar dois cálculos: o cálculo de ativação, que gera o valor de saída de cada neurônio e o cálculo de uma estimativa do vetor gradiente, que é necessário para a retropropagação através da rede.

As redes MLP apresentam um poder computacional muito maior do que aquele apresentado pelas redes sem camadas ocultas, podendo trabalhar com dados que não são linearmente separáveis. Teoricamente, redes com duas camadas ocultas podem implementar qualquer função, seja ela linearmente separável ou não [19].

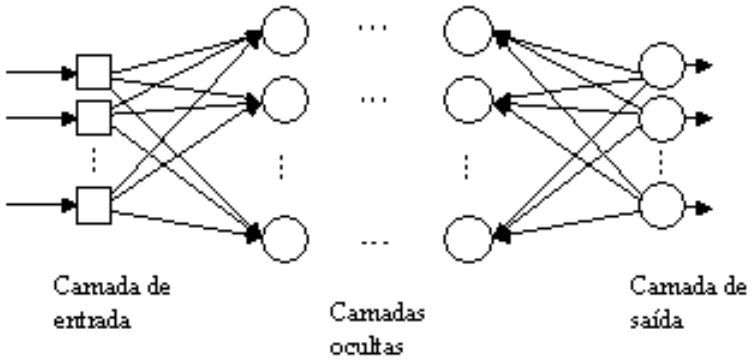

FIGURA 2.1 – Representação gráfica da arquitetura de uma rede MLP, com uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída, onde cada neurônio de cada camada da rede está conectado a todos os neurônios da camada anterior.

Um dos principais aspectos relacionados ao projeto de redes MLP diz respeito a função de ativação utilizada. Diversas funções de ativação têm sido propostas para redes multicamadas, sendo estas funções não-lineares e diferenciáveis, para que o gradiente possa ser calculado, direcionando o ajuste dos pesos. Entre as função de ativação encontradas, uma das mais utilizadas é a função sigmóide logística, definida por

$$y_j = \frac{1}{1 + exp(-v_j)}, \tag{2.1}$$

onde v_j é a soma das ativações dos neurônios de entrada aplicadas ao neurônio j acrescidas do bias, e y_j é a saída do neurônio.

2.1.2 Funcionalidade

Em uma rede multicamadas, o processamento realizado por cada neurônio é definido pela combinação dos processamentos realizados pelos neurônios da camada anterior que estão conectados a ele. Quando segue da primeira camada intermediária em direção a camada de saída, as funções implementadas se tornam cada vez mais complexas. Estas funções definem como é realizada a divisão do espaço de decisão.

Pode ser dito que os neurônios intermediários de uma rede MLP funcionam como detectores de características. Elas geram uma codificação interna dos padrões de entrada, que é então utilizada para a definição da saída da rede. Dado um número suficientemente grande de unidades intermediárias, é possível formar representações internas para qualquer conjunto de padrões de entrada.

Cybenko [19], além de outros pesquisadores [41], [28], investigou o número de camadas intermediárias necessárias para a implementação de classes de funções em uma RNA. Os resultados obtidos indicam que:

- Uma camada intermediária é suficiente para aproximar qualquer função contínua [19], [36].

- Duas camadas intermediárias são suficientes para aproximar qualquer função matemática [18].

O número de neurônios das camadas ocultas é em geral definido empiricamente. Este número depende fortemente da distribuição dos padrões de treinamento e de validação da rede.

2.1.3 Treinamento de Redes MLP

Atualmente, existem vários algoritmos para treinar redes MLP [70], [25], [68], [6], [32]. Estes algoritmos são geralmente supervisionados, e podem ser classificados como estáticos ou dinâmicos, de acordo com os parâmetros que eles atualizam. Enquanto os algoritmos estáticos não alteram a estrutura da rede, variando apenas os valores de seus pesos, os algoritmos dinâmicos podem mudar a estrutura da rede. Esta mudança pode envolver a eliminação ou acréscimo de neurônios e/ou conexões da rede, ou uma combinação destes dois processos. Uma rede construtiva que tem sido amplamente utilizada é a rede *Cascade Correlation* [26]. O algoritmo de aprendizado mais conhecido para o treinamento destas redes é o algoritmo de retropropagação (*back-propagation*) [70], sendo que a maioria dos métodos utilizados para o mesmo fim, utiliza variações deste algoritmo.

O algoritmo de retropropagação é um algoritmo supervisionado que utiliza pares de entrada e saída desejada para, por meio de um mecanismo de correção de erros, ajustar os pesos da rede. O treinamento ocorre em duas fases chamadas propagação e retropropagação. A fase de propagação é utilizada para definir a saída da rede neural para um dado padrão de entrada. A fase de retropropagação utiliza a saída desejada e a saída fornecida pela rede para atualizar os pesos de suas conexões. Durante o passo de propagação, os pesos sinápticos da rede são todos

fixos, por outro lado, na retropropagação os pesos sinápticos são todos ajustados de acordo com uma regra de correção de erro, onde a resposta real da rede é subtraída de uma resposta desejada para produzir um sinal de erro. Este sinal de erro é então propagado para trás através da rede, contra a direção das conexões sinápticas. Utilizando o método do gradiente descendente, o algoritmo de retropropagação procura minimizar o erro obtido pela rede ajustando os pesos e limiares para que eles correspondam às coordenadas dos pontos mais baixos da superfície de erro.

Desde a sua criação, várias alterações do algoritmo *back-propagation* têm sido propostas visando tanto a acelerar seu tempo de treinamento como a melhorar seu desempenho. Destas variações, as mais utilizadas são *Back-propagation* com *momentum* [70], Quickprop [25], Levenberg-Marquardt [32], *momentum* de segunda ordem [66], Newton [6] e Rprop [68].

2.2 Sistemas Dinâmicos

Nesta seção são descritos conceitos sobre a teoria de sistemas dinâmicos, focando principalmente características apresentadas por sistemas não-lineares e sistemas caóticos. Um sistema é uma combinação de componentes que atuam em conjunto para satisfazer um objetivo especificado. Um sistema é estático quando sua saída atual depende somente da entrada atual, a saída do sistema só varia se a sua entrada variar. A definição mais simples para sistemas dinâmicos é a de sistemas que variam no tempo, sendo, sua principal característica, a utilização da variável tempo nas equações matemáticas que os representam. A teoria de sistema dinâmicos se ocupa em descrever matematicamente sistemas em movimento, permitindo classificar e predizer seu comportamento no tempo.

O modelo matemático de um sistema dinâmico é definido como sendo o conjunto de equações que representam a dinâmica do sistema com uma certa precisão, baseando-se nas leis que regem o sistema em questão. O modelo matemático de um dado sistema não é único, um sistema pode ser representado por diferentes modelos dependendo da análise que se deseja fazer. Quanto mais simples o modelo matemático obtido para um dado sistema, menor a precisão apresentada pelo modelo em comparação ao sistema real. O modelo é então tanto melhor quanto maior for sua fidelidade em relação ao sistema que modela [21]. De posse de um modelo matemático, pode-se executar a simulação do sistema a que o modelo concerne, isto é, usá-lo para obter previsões dos estados futuros do sistema modelado a partir de seus estados em momentos anteriores.

Os sitemas dinâmicos podem ser divididos em determinísticos e estocásticos. Os sistemas determinísticos são aqueles regidos por equações determinístcas enquanto que os estocásticos só podem ser estudados por suas propriedades probabilísticas. Os sistemas dinâmicos podem ser ainda divididos em discretos e contínuos. Em sistemas discretos, a variável temporal assume valores de forma discreta, uma aproximação válida em casos onde a escala temporal típica de variações é muito maior que o intervalo de discretização. Os sistemas dinâmicos contínuos são aqueles em que a variável temporal assume valores de forma contínua. Os exemplos mais típicos são os sistemas diferenciais, isto é, aqueles representados através de um sistema de equações diferenciais ordinárias [37].

2.2.1 Sistemas Não-Lineares

Quando um sistema dinâmico não apresenta a propriedade de linearidade ele é denominado sistema dinâmico não-linear. No entanto, quando a faixa de operação do sistema é pequena e as não-linearidades são suaves, um sistema dinâmico não-linear pode ser representado aproximadamente por seu correspondente sistema linearizado cuja dinâmica é descrita por um conjunto de equações lineares [44].

Um sistema linear é um sistema no qual as alterações em um estado inicial resultarão em alterações proporcionais em qualquer estado subseqüente. Já um sistema não-linear, é aquele no qual as alterações em um estado inicial não necessitam produzir alterações proporcionais em estados subseqüentes [52].

No modelo de espaço de estados, um conjunto de variáveis de estado assume valores (em um instante particular de tempo) que contem informação suficiente para prever a evolução futura do sistema, levando em consideração as leis que regem aquele sistema. A dinâmica de uma grande classe de sistemas autônomos não-lineares pode ser especificada na forma de um sistema de equações diferenciais de primeira ordem:

$$\frac{d}{dt}x_j(t) = F_j(x_1(t), ..., x_N(t)) \tag{2.2}$$

onde a função F_j é, em geral, uma função não-linear.

Utilizando-se a notação vetorial para o sistema de equações, tem-se:

$$\frac{d}{dt}\vec{x}(t) = \vec{F}(\vec{x}(t)) \tag{2.3}$$

onde a função não-linear F tem valor vetorial, com cada um de seus elementos operando sobre um elemento correspondente do vetor de estado $\vec{x}(t)$, Eq. (2.4), que deve variar com o tempo t de acordo com F.

$$\vec{x}(t) = [x_1(t), x_2(t), ..., x_N(t)]^T \tag{2.4}$$

O espaço de estados fornece uma ferramenta conceitual para analisar a dinâmica de um sistema não-linear. No espaço de estados a atenção é focada sobre as características globais do movimento em vez de fixar nos aspectos detalhados de soluções analíticas ou numéricas da equação [62].

Em um instante particular de tempo t, o estado $x(t)$ observado do sistema é representado por um único ponto no espaço de estados N-dimensional. As mudanças no estado do sistema são representadas com uma curva no espaço de estados, que é chamada de trajetória ou órbita do sistema. A família de trajetórias, para diferentes condições iniciais, é referida como o retrato de estados do sistema, que inclui todos os pontos no espaço onde o campo vetorial $F(x)$ é definido. Para um sistema autônomo haverá apenas uma trajetória passando através de um estado inicial. Do retrato de estados, obtém-se o fluxo de um sistema dinâmico, definido como o movimento do espaço de estados dentro dele mesmo.

Dado um retrato de estados de um sistema dinâmico, pode-se construir um campo de vetores velocidade, um para cada ponto do espaço de estados. Um campo vetorial dá uma

descrição visual da tendência inerente de um sistema dinâmico de se mover com uma velocidade habitual em cada ponto específico de um espaço de estados. A análise de sistemas dinâmicos via estruturas de espaço de estados possui um importante papel em experimentação e interpretação de sistemas dinâmicos.

O espaço de estados de um sistema dinâmico não-linear seguidamente consiste de regiões qualitativamente diferentes, sendo útil examinar as informações geométricas sobre as estruturas e arranjos espaciais dessas regiões. Entretanto as características mais importantes de um sistema dinâmico são os pontos fixos, órbitas periódicas, seus tipos de estabilidade e o arranjo espacial das regiões de estabilidade correspondentes [80].

A análise de sistemas dinâmicos não-lineares, em termos das características topológicas de seus atratores, é conhecida como análise qualitativa. Através dessa análise é possível estimar as invariantes dinâmicas não-lineares, como: Dimensão, que fornece uma medida dos graus de liberdade de um sistema, para que este possa ser considerado previsível, este valor deve ser pequeno, para sistemas que apresentam dimensões altas, uma modelagem randômica é mais apropriada; Entropia, que fornece a taxa de perda de informação sobre as condições iniciais do sistema durante sua evolução, sendo que a previsibilidade temporal de um sistema é inversamente proporcional a sua entropia; e Expoentes de Lyapunov, que medem a estabilidade local de uma trajetória no espaço de fases, para espaços de fases m-dimensionais, m expoentes de Lyapunov podem ser extraídos [1].

2.2.2 Sistemas Caóticos

Uma das formas de um modelo matemático apresentar complexidade intrínseca é através de certos regimes, denominados de caóticos. Nos regimes caóticos, os modelos apresentam sensibilidade às condições iniciais, o que faz com que os erros na determinação das condições iniciais ou nos cálculos envolvidos na definição dos modelos se amplifiquem fortemente. Esta propagação de erros compromete, na prática, qualquer previsão baseada neste modelo, quebrando o paradigma determinista de previsibilidade.

A impossibilidade de predizer por que ponto do espaço de fases a trajetória de um atrator estranho passará num certo instante de tempo é uma característica intrínseca de todos os sistemas caóticos. No entanto, ainda pode-se fazer previsões muito precisas, mas elas se referem às características qualitativas do comportamento do sistema, e não aos valores precisos de seu estado num determinado instante.

A teoria do caos não é uma teoria de desordem, mas busca no aparente acaso uma ordem intrínseca determinada por leis precisas. Trata-se de um comportamento dinâmico não-linear, determinístico, mas imprevisível a longo prazo. Além do clima, outros processos aparentemente casuais apresentam certa ordem, como por exemplo o crescimento populacional, arritmias cardíacas, flutuação do mercado financeiro, etc. [37].

Sistemas dinâmicos lineares não podem exibir comportamento caótico, ou seja, atratores ou repulsores caóticos só ocorrem no caso de sistemas dinâmicos não-lineares. Dinâmicas lineares podem apenas expandir, comprimir ou rotacionar o espaço de estados, de forma que apenas pontos fixos e ciclos periódicos são possíveis. À primeira vista, a dinâmica caótica se assemelha a muitos processos aleatórios, parecendo não seguir nenhum tipo de padrão. No entanto, sistemas

caóticos são descritos por equações determinísticas, enquanto que um processo verdadeiramente aleatório pode ser caracterizado apenas em termos de propriedades estatísticas [62].

Uma descrição matemática para sistemas caóticos discretos dada por Devaney [21], é que um mapa $f : V \rightarrow V$ é dito caótico se: f possui sensibilidade às condições iniciais, que implicará em imprevisibilidade; f é topologicamente transitivo, independente da condição inicial escolhida, cedo ou tarde, todos os estados possíveis serão aproximadamente atingidos; e o conjunto dos pontos periódicos de f é denso em V, o que acarreta uma certa regularidade implícita no mesmo, implicando num comportamento de retorno a todos os estados possíveis .

O objeto geométrico no espaço de estados para o qual uma trajetória caótica é atraída é denominado atrator estranho por possuir dimensão fracionária decorrente de sua geometria fractal. Da mesma forma que atratores periódicos podem ser diferenciados por seus períodos, atratores estranhos podem ser diferenciados por sua dimensão. Pontos fixos e ciclos limites de tempo discreto são atratores de dimensão zero, pois são constituídos apenas por um número finito de pontos. Ciclos limites de tempo contínuo são atratores de dimensão inteira. Por exemplo, quando o ciclo limite é uma curva que se liga em suas extremidades, sua dimensão é um. Já atratores caóticos podem apresentar dimensão fracionária.

Para sistemas dinâmicos, os expoentes de Lyapunov fornecem uma medida de sensibilidade do sistema às condições iniciais. Quando o atrator associado ao sistema é caótico, ao considerar-se duas trajetórias cujas condições iniciais são muito próximas, as trajetórias divergem, em média, a uma taxa exponencial caracterizada pelo maior expoente de Lyapunov. A presença de um expoente positivo é suficiente para garantir que as órbitas são exponencialmente sensíveis às condições iniciais ou a perturbações, diagnosticando-se o caos. O atrator de um sistema dissipativo caótico, isto é, com um ou mais expoentes de Lyaponuv positivos, é classificado como estranho.

2.2.3 Redes Neurais Recorrentes como Sistemas Dinâmicos Não-Lineares

A incorporação de um princípio físico fundamental, a armazenagem de informação em uma configuração dinamicamente estável, ou seja, a memória, caracteriza as Redes Neurais Recorrentes como um sistema dinâmico não-linear. Cada padrão a ser armazenado fica localizado em um vale da superfície de energia. Como a dinâmica não-linear da rede é estabelecida de modo a minimizar a energia, os vales representam pontos de equilíbrio estáveis (cada qual com a sua base de atração).

Redes Neurais Recorrentes podem ser visualizadas como sistemas dinâmicos de tempo discreto. Em Tiño *et al.* [80] é mostrado que Redes Neurais com apenas dois neurônios recorrentes podem exibir caos e a partir disso, um comportamento dinâmico da rede (em um atrator caótico) pode ser muito complexo. Embora outros pesquisadores já buscassem a implementação do conceito de memória como ponto de equilíbrio estável, Hopfield [39] foi o primeiro a formulá-lo em termos precisos. Este tipo de sistema dinâmico pode operar como: memória associativa (endereçável por conteúdo) e dispositivo computacional para resolver problemas de otimização de natureza combinatória.

A memória endereçável por conteúdo, tem como principal função restaurar um padrão binário armazenado (item de memória), em resposta à apresentação de uma versão incompleta

(papel restaurador) ou ruidosa (papel de corretor de erro) deste padrão. A essência da memória endereçável por conteúdo é mapear uma memória fundamental em um ponto fixo estável do sistema dinâmico representado pela Rede Recorrente. Logo, a Rede Neural de Hopfield é um sistema dinâmico não-linear cujo espaço de estados contém um conjunto de pontos fixos estáveis que representam as memórias fundamentais do sistema.

2.3 Autômatos Finitos Determinísticos

Esta seção apresenta conceitos sobre os Autômatos Finitos (AF), que estão entre os modelos fundamentais na ciência da computação e na teoria de circuitos seqüênciais. Eles modelam sistemas dinâmicos discretos gerais. Um autômato de estados finitos é um modelo matemático de um sistema com entradas e saídas discretas, que pode assumir um número finito e pré-definido de estados. Cada estado resume somente as informações do passado necessárias para determinar as ações para a próxima entrada [38].

Dois AF são considerados equivalentes se produzirem, para cada possível seqüência de entrada, igual seqüência de saída. Um AF é considerado mínimo se não existir autômato equivalente com menor número de estados. Ele opera em um regime esperado somente se entradas definidas são aplicadas durante sua operação, que é descrita pela função de transição de estados e pela função de saída.

O estado de um sistema é a descrição do sistema em um determinado instante. Ele fornece todas as informações necessárias para determinar como o sistema pode evoluir a partir daquele ponto. As mudanças de estado realizadas espontaneamente, em resposta a uma entrada externa ou instantânea, são denominadas transições, sendo, estas mudanças, gerenciadas pela função de transição de estados. O autômato pode ser visualizado, através da utilização de uma ferramenta gráfica chamada diagrama de transição de estados definida por um grafo finito direcionado que representa a função de transição do autômato.

Um Autômato Finito Determinístico (AFDet) é aquele em que, uma transição iniciada em um determinado estado, leva a um único outro estado. No entanto, em um Autômato Finito Não-Determinístico (AFND), podem existir transições que levam de um determinado estado para vários outros estados do autômato. O AFDet é um modelo de sistema dinâmico em que existe um número finito de estados do sistema, existem transições entre estados mediadas por entradas provenientes de um conjunto discreto finito e para cada estado, todas as transições produzidas por cada um dos símbolos do alfabeto estão definidas [13].

Um AFDet pode ser definido por uma 6-tupla $M = (X, S, Y, f_s, f_o, s_0)$, onde X é um conjunto de entrada finito não vazio, S é um conjunto não vazio de estados internos, Y é um conjunto finito não vazio de saída, f_s denota a função de transição de estados $s(t+1) = f_s(x(t), s(t))$, $f_s : X * S \rightarrow S$, f_o denota a função de saída $y(t) = f_o(s(t))$, $f_o : S \rightarrow Y$ *e* s_0 é o estado inicial, pertencente ao conjunto dos estados internos.

2.4 Cadeias de Markov

Nesta seção são apresentados alguns conceitos sobre processos estocásticos e, mais precisamente, sobre cadeias de Markov. Um processo estocástico é uma coleção de variáveis aleatórias indexadas por um parâmetro de tempo n, definidas em um espaço denominado espaço de estados. O valor x_n assumido pela variável aleatória X_n em um determinado instante de tempo é chamado de estado. Um processo aleatório, X_n, é um processo de Markov se o futuro do processo dado o presente é independente do passado.

Em uma cadeia de Markov, o estado atual depende unicamente do estado predecessor, como pode ser visto na Eq. (2.5):

$$P[X_{n+1} = x_{n+1}|X_n = x_n, ..., X_1 = x_1] = P[X_{n+1} = x_{n+1}|X_n = x_n]. \tag{2.5}$$

Ou seja, uma seqüência de variáveis aleatórias X_1, X_2,..., X_n, X_{n+1} forma uma cadeia de Markov se a probabilidade de que o sistema esteja no estado x_{n+1} no tempo n+1 depende exclusivamente da probabilidade de que o sistema esteja no estado x_n no tempo n *[34]*. A Figura 2.2 apresenta um modelo básico de cadeia de Markov, onde os estados são denominados como S_1, S_2 e S_3 e cada transição possui uma função de probabilidade associada.

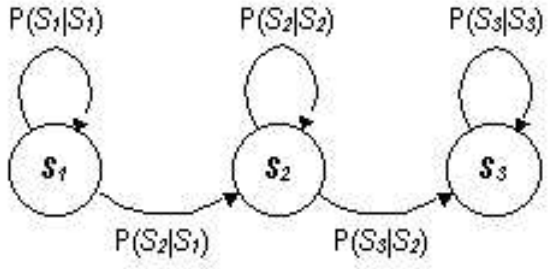

FIGURA 2.2 – Visualização em forma de grafo de uma Cadeia de Markov.

Em uma cadeia de Markov, a transição de um estado para outro é probabilística, mas a produção de um símbolo de saída é determinística. A probabilidade de transição de um estado i no tempo n para o estado j no tempo n+1, é dada por

$$p_{ij} = P(X_{n+1} = x_j|X_n = x_i) \tag{2.6}$$

Todas as probabilidades de transição devem satisfazer as seguintes condições:

$$p_{ij} \geq 0 \text{ para todo } (i, j) \tag{2.7}$$

e

$$\sum_j p_{ij} = 1 \text{ para todo } i. \tag{2.8}$$

No caso de um sistema com um número finito K de estados possíveis, as probabilidades de transição constituem uma matriz $\boldsymbol{P}$, K-por-K, chamada de matriz estocástica, cujos elementos individuais satisfazem as condições descritas nas Eqs. (2.7) e (2.8), onde a soma de cada linha de $\boldsymbol{P}$, mostrada na Eq. (2.9) , deve resultar em 1.

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1K} \\ p_{21} & p_{22} & \cdots & p_{2K} \\ \vdots & \vdots & & \vdots \\ p_{K1} & p_{K2} & \cdots & p_{KK} \end{bmatrix} \tag{2.9}$$

Generalizando a definição de probabilidade de transição dada na Eq. (2.6) para casos em que a transição de um estado para o outro ocorra em um número fixo de passos, obtêm-se

$$p_{ij}^{(m)} = P(X_{n+m} = x_j | X_n = x_i), \quad m = 1, 2, ... \tag{2.10}$$

onde $p_{ij}^{(m)}$ representa a probabilidade de transição em m passos do estado i para o estado j.

O processo descrito é denominado modelo observável de Markov [53], sendo que a saída do processo é o conjunto de estados em cada instante de tempo, onde cada estado corresponde a um evento físico observável. Quando a observação é uma função probabilística do estado e não um evento determinístico, é necessário estender o conceito de modelos de Markov para um modelo chamado Modelo Oculto de Markov (*Hidden Markov Model* - HMM), que é um processo estocástico oculto que pode ser observado unicamente através de outro processo estocástico que produz a seqüência de observações.

HMM's são máquinas de estados finitas estocásticas constituídas por um conjunto de estados ligados entre si através de transições [67]. Eles podem ser utilizados para a representação de processos dinâmicos. Os HMM são especificados por um conjunto de estados, um conjunto de probabilidades de transição para cada transição permitida entre estados, e uma função de densidade de probabilidade associada com a emissão de símbolos.

Em um modelo de Markov comum (não oculto), pode-se observar a seqüência de estados percorrida pelo modelo, pois existe uma relação direta entre o símbolo emitido e o estado em que se encontra o modelo. Em outras palavras, cada estado pode emitir apenas um símbolo, que representa o estado em que se encontra o sistema. Já nos HMMs, cada estado pode emitir qualquer símbolo de acordo com uma distribuição de probabilidades.

2.5 Lógica Difusa

Esta seção apresenta conceitos de lógica difusa e fundamentos da teoria de conjuntos difusos introduzida por Lotfi A. Zadeh em 1965 [89]. Esta teoria foi desenvolvida para tratar de problemas demasiado complexos ou mal definidos para serem tratados pelos métodos matemáticos clássicos [58]. Zadeh observou que os recursos tecnológicos disponíveis eram incapazes de dar resposta a alguns dos problemas, da atividade industrial, da biologia, da química, que comportassem situações ambíguas, não passíveis de processamento através da lógica computacional.

A Lógica Difusa encontra-se entre as técnicas mais recentes da Inteligência Artificial. Ela surgiu, da necessidade de aproximar a decisão computacional da decisão humana e desenvolveu a habilidade de extrair conclusões e gerar respostas baseadas em informação vaga, ambígua, incompleta. Os parâmetros de sistemas difusos adaptativos têm clareado significados físicos que facilitam a escolha de seus valores iniciais.

O aspecto mais notável dessa metodologia é a possibilidade de se capturar, em um modelo matemático, conceitos intuitivos como graus de satisfação, conforto, adequação, etc. Toda a experiência do especialista pode ser capturada, transformando subjetividade em graus de pertinência, raciocínio em base de regras, tomada de decisão em inferência/*defuzzificação*. Ao invés de variáveis numéricas, a Lógica Difusa utiliza variáveis lingüísticas que admitem como valores, apenas, expressões lingüísticas (ou termos primários), como "muito grande", "pouco frio", "mais ou menos jovem", que são representadas por conjuntos difusos. A estratégia de controle de um operador humano pode ser representada como um conjunto de relações condicionais difusas que formam um conjunto de regras de decisão.

A combinação de Redes Neurais Artificiais com lógica difusa tem sido utilizada em muitos domínios de aplicação. Modelos que utilizam Redes Neurais e lógica difusa (*neurofuzzy*) aumentam o poder de representação para aplicações que requerem contexto e estado, como por exemplo, em reconhecimento de fala, predição de séries temporais e controle.

2.5.1 Conjuntos Difusos

Na teoria clássica dos conjuntos, utiliza-se como princípio básico a idéia de que um elemento "pertence" ou "não pertence" a um determinado conjunto, restringindo as fronteiras dos conjuntos e dando o mesmo peso à diferentes objetos que, de uma certa maneira, poderiam ter mais afinidades em um determinado conjunto, do que em outro.

De acordo com Zadeh, o termo difuso é usado em situações em que um conjunto A, definido sob um universo X, não apresenta seus limites bem definidos. Para aqueles elementos que com certeza pertencem ao conjunto A, é atribuído um grau de pertinência igual a 1. Para os elementos que com certeza não pertencem ao conjunto A, é atribuído um grau de pertinência igual a zero. Para os elementos os quais não se pode afirmar com certeza se pertencem ou não pertencem ao conjunto A, é atribuído um valor intermediário, tendendo para 1, quanto maiores forem as razões que se tem para incluir este elemento no conjunto A [45].

Segundo Zadeh, o conjunto difuso é definido como sendo uma classe de objetos com contínuos graus de pertinência, a qual designa a cada objeto um grau de pertinência, que está no intervalo [0,1]. Isto significa que: sendo X um conjunto clássico de objetos chamado universo, cujos elementos genéricos são denotados por x, então a função de pertinência de um elemento x em um subconjunto clássico é uma função característica $\mu_{\mathbf{A}}(x)$, com X -> {0,1}, definido como conjunto de avaliação. Neste caso, A é um conjunto difuso e $\mu_{\mathbf{A}}(x)$ representa o grau de pertinência de x em relação a A , desde que A seja o subconjunto de X que não tem fronteira bem definida.

É importante trabalhar com conjuntos difusos normalizados, ou seja, que apresentem altura unitária. Os contornos de um conjunto difuso representam as propreidades semânticas do conceito subjacente. Logo, quanto mais próxima a curva estiver do comportamento do fenômeno em

estudo, melhor será o desempenho do modelo difuso global em seu papel de representar com precisão o mundo real. Contudo, os modelos difusos são tolerantes a aproximações relativamente grosseiras. Pode-se ter um bom desempenho mesmo onde não se tenha conseguido descrever com excelente precisão o comportamento dos objetos sob análise.

Na síntese de conjuntos difusos, tem-se extrema flexibilidade na escolha da forma geométrica da função que descreve os vários graus de pertinência dos elementos do universo de discurso em relação a um dado subconjunto difuso. As funções mais encontradas na prática são triangulares, trapezoidais, gaussianas e sigmóides (conforme Figura 2.3), sendo normalmente utilizadas 3, 5 ou 7 áreas. As triangulares e gaussianas aparecem normalmente em casos nos quais se deseja exprimir pertinência crescente à esquerda e decrescente à direita. As funções trapezoidais podem ser usadas em situações similares, em que se queira "alargar" a faixa de pertinência máxima. As curvas sigmóides são usualmente aplicadas em casos em que se busca delimitar casos extremos, a partir dos quais a pertinência se mostra constante [45].

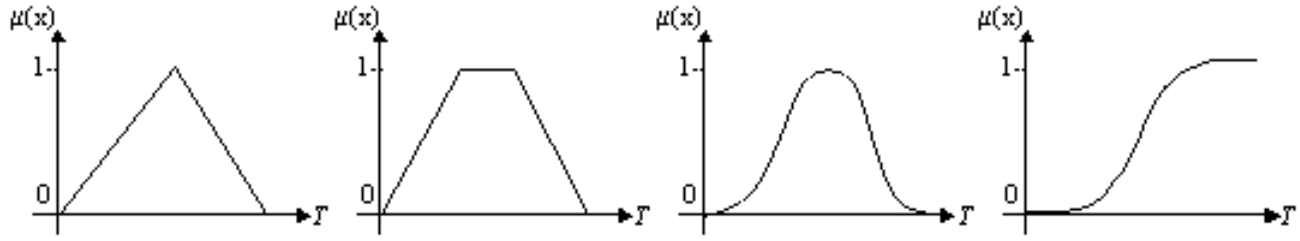

FIGURA 2.3 – Representações gráficas das formas encontradas para as funções de pertinência.

2.5.2 Operações com Conjuntos Difusos

Uma das consequências naturais à generalização da teoria clássica dos conjuntos para a teoria dos conjuntos difusos é o estudo das operações sobre conjuntos. O componente crucial de um conjunto difuso é sua função de pertinência, assim, as operações sobre conjuntos difusos são definidas via funções de pertinência. A seguir caracteriza-se algumas operações com conjuntos difusos dados pelo princípio da extensão proposto por Zadeh, que é utilizado para estender os conceitos, rigor e formalismo da matemática clássica à teoria dos conjuntos difusos [45].

Na teoria clássica dos conjuntos, a intersecção entre dois conjuntos, contém aqueles elementos que são comuns a ambos. Na teoria dos conjuntos difusos entretanto, o elemento pode pertencer parcialmente aos dois conjuntos, ainda que não pertença completamente a nenhum deles. Assim, quando é considerada a intersecção desses conjuntos, não se pode dizer que um elemento possa pertencer mais ao conjunto da intersecção do que a qualquer um dos conjuntos originais. De acordo com isto, o operador difuso padrão usado para gerar a intersecção entre dois conjuntos difusos A e B definidos em X é dada por:

$$\mu_A(x) \cap \mu_B(x) = min(\mu_A(x), \mu_B(x)) = \mu_A(x) \wedge \mu_B(x), \forall x \in X \tag{2.11}$$

onde "$\wedge$" ("E" lógico) é usado na lógica difusa para representar o operador *min*, que toma o mínimo entre os valores em consideração.

Outra forma de combinar conjuntos difusos é através de sua união. A união de dois conjuntos é compreendida como sendo o conjunto dos elementos que pertencem a pelo menos um deles (ou a ambos). Assim sendo os elementos do conjunto união não podem possuir valor de pertinência menor que o que possuía em qualquer um dos conjuntos originais. A forma padrão que a lógica difusa usa para obter a união entre dois conjuntos é a seguinte:

$$\mu_A(x) \cup \mu_B(x) = max(\mu_A(x), \mu_B(x)) = \mu_A(x) \vee \mu_B(x), \forall x \in X \tag{2.12}$$

onde "$\vee$" ("OU" lógico) é utilizado na lógica difusa para representar a operação *max*, que toma o valor máximo dentre os valores em consideração.

Dado um conjunto difuso A, é possível encontrar o seu complemento $\tilde{}A$ através da seguinte expressão:

$$\mu_A(x) = 1 - \mu_A(x) \tag{2.13}$$

As operações de união e intersecção com conjuntos difusos, descritas acima, podem ser visualizadas na Figura 2.4.

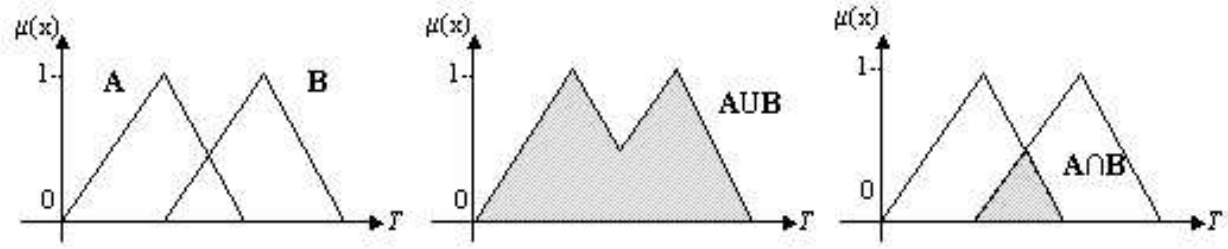

FIGURA 2.4 – Representações gráficas das operações de união e intersecção utilizando dois subconjuntos A e B.

Capítulo 3

Redes Neurais Temporais

3.1 Introdução

Este capítulo trata das Redes Neurais Temporais, encontradas em um amplo número de trabalhos. São apresentadas algumas das arquiteturas encontradas para este tipo de rede, juntamente com alguns algoritmos de aprendizagem utilizados para o seu treinamento. A última seção é dedicada ao problema de extração de conhecimento a partir de Redes Neurais Temporais.

Muitos algoritmos de treinamento das RNAs não são capazes de implementar mapeamentos dinâmicos, como por exemplo o algoritmo *back-propagation*, que pode ser usado apenas para mapeamentos estáticos. Um artifício utilizado para processamento temporal envolve o uso de janelas de tempo, em que a entrada da rede utiliza trechos dos dados temporais como se eles formassem um padrão estático.

É possível estender a estrutura das redes MLP para que assumam um comportamento que varie com o tempo, tornando-as capazes de tratar sinais temporais. O tempo pode ser representado pelo efeito que acarreta no processamento de sinais, que significa oferecer características dinâmicas ao mapeamento realizado pela rede, tornando-a sensível a sinais que variam com o tempo.

Entre as Redes Neurais convencionais, o uso da MLP com o algoritmo de aprendizado de retropropagação demonstrou capacidade de realizar mapeamentos dinâmicos. Entretanto, muitos pesquisadores dedicaram-se a encontrar arquiteturas de RNAs adequadas para este tipo de mapeamento. Para que uma Rede Neural seja considerada dinâmica, ela deve possuir memória [23]. Isto é possível se a rede considerar entradas atrasadas no tempo, ou se a rede tiver laços de realimentação.

Fazendo analogia à circuitos seqüênciais, as Redes Neurais Temporais podem ser visualizadas conforme o modelo geral mostrado na Figura 3.1. Para a obtenção de processamento temporal, uma possibilidade é apresentar como entradas, variáveis externas e variáveis de estado, sendo estas realimentadas do processamento de saída para o processamento de entrada. Outra forma é possuir a memória como entrada da rede, assim, o processamento de entrada gera os valores das variáveis de estado para o próximo passo de tempo, e estes valores são apresentados como entrada para o processamento de saída.

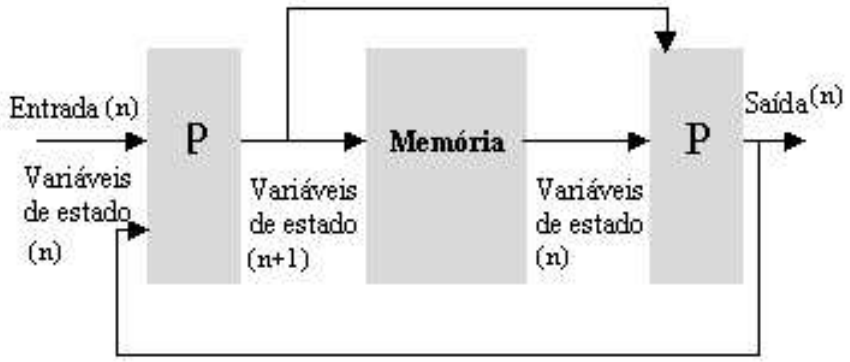

FIGURA 3.1 – Representação de um modelo geral de circuito seqüêncial, apresentando as características de uma Rede Neural Temporal.

As RNA Temporais, em função de sua estrutura, consideram o tempo na sua operação, incorporando memória de curto prazo distribuída na rede em todos os neurônios escondidos e em alguns dos casos nos neurônios de saída. Esta classe de redes é utilizada para representar melhor a natureza temporal dos sistemas dinâmicos.

O tempo constitui um ingrediente essencial no processo de aprendizagem. Ele pode ser contínuo ou discreto, sendo, independente de sua forma, uma entidade ordenada que é básica para muitas tarefas cognitivas encontradas na prática, como a visão, a fala, o processamento de sinais e o controle motor. Para a incorporação do tempo na operação de uma Rede Neural existem duas possibilidades [34]: representação implícita e representação explícita.

Na representação implícita, o tempo é representado pelo efeito que tem sobre o processamento de sinais de uma maneira implícita. Uma rede neural estática (por exemplo uma *perceptron* de múltiplas camadas) é suprida com propriedades dinâmicas, que, por sua vez, torna a rede sensível à estrutura temporal dos sinais portadores de informação. Neste tipo de representação a memória é provida através da introdução de atrasos no tempo, como pode ser observado nas técnicas TDNN *Time Delay Neural Networks* [51], [82], [83] e FIR *Multilayer Perceptron* [84].

A outra forma de incorporação do tempo na operação de uma Rede Neural é através da representação explícita, onde o tempo recebe sua própria representação particular, ou seja, prover memória utilizando Redes Recorrentes, tais como *Back-propagation Through Time* [85], [87], *Real-Time Recurrent Learning* [88], *Cascade Correlation* Recorrente [24], redes de Elman [23] e redes de Jordan [42].

Redes Neurais Temporais são usadas em importantes problemas atuais como na previsão e modelagem de séries temporais [33], no processamento de sons (voz) [71] e de imagens, na previsão de indicadores financeiros [72], na modelagem de sistemas caóticos [81]; no cancelamento de ruído [30]; no controle adaptativo [47]; controle adaptativo em robótica [2], [57], [64], [78]; na identificação de sistemas [56], [43], eliminação de ruídos e de eco, equalização adaptativa, correção de não-linearidades em amplificadores; na simulação numérica e visualização [31]; aprendizado de seqüências (no planejamento, robótica, processamento de linguagem, sequenciamento de DNA, proteínas, etc.) [76], aprendizado de estruturas sintáticas [15], etc.

3.2 Redes Neurais Alimentadas Adiante com Atrasos no Tempo

Nesta seção são apresentadas, de forma bastante sucinta, as Redes Neuaris Alimentadas Adiante com Atrasos no Tempo. A idéia básica do processamento espaço-temporal utilizando atrasos no tempo é mostrada na Figura 3.2, onde a resposta da RNA típica *(feedforward)*, no tempo n, é baseada nas entradas no tempo *(n-1), (n-2),..., (n-q)*. Desta forma, considera-se um histórico da seqüência temporal. Esta arquitetura, onde os atrasos são utilizados somente na camada de entrada, é muito utilizada, mas existem generalizações do método para considerar atrasos também nas camadas ocultas e de saída, o que possibilita melhorar consideravelmente o desempenho, mas acarreta também no aumento de complexidade para o uso da rede.

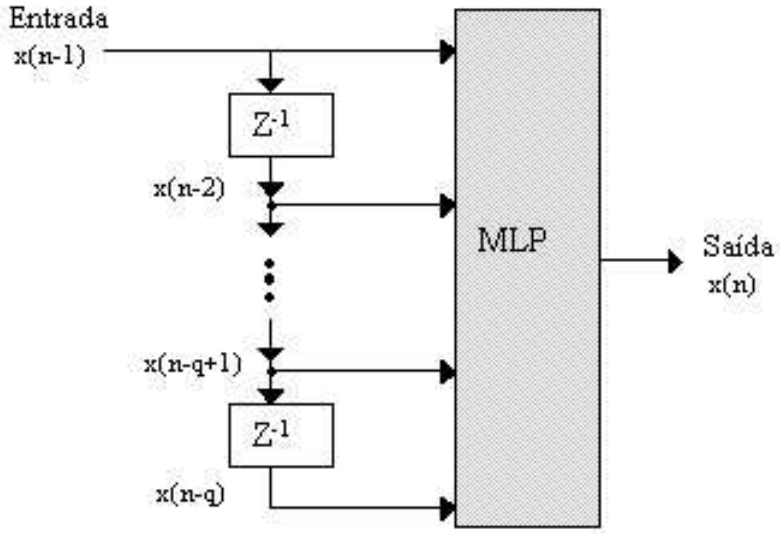

FIGURA 3.2 – Processamento temporal usando atrasos no tempo.

Este tipo de RNA apresenta como entrada, os valores das variáveis de estado atrasadas no tempo, ou seja, a memória é apresentada como entrada para o processamento de saída (Veja a Figura 3.3).

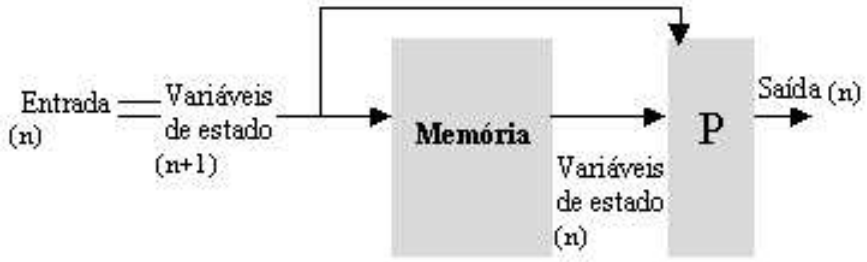

FIGURA 3.3 – Representação das características de uma RNA com processamento temporal utilizando atrasos no tempo, através de um modelo de circuito seqüêncial.

Os modelos de RNAs mais utilizados para processamento temporal foram, por muito tempo, as redes com atraso no tempo. A idéia sempre foi introduzir memória à rede proporcionando aos neurônios valores de entrada atuais e valores temporalmente anteriores a eles. A Figura 3.4 ilustra uma rede com duas camadas escondidas utilizando o método de janelamento. Os neurônios da primeira camada escondida além de receberem a entrada atual *x(n)*, recebem também, neste caso, as duas entradas anteriores: *x(n-1)* e *x(n-2)*; criando-se, porem, 8 sinapses novas, para este exemplo. Isto é, o método de janelamento proporcionou à rede uma memória de ordem 2 só na primeira camada escondida. Mas ao final, para o processo de aprendizado e teste da rede, esta é considerada simplesmente como um caso de rede estática padrão totalmente interconectada (entre os neurônios de camadas adjacentes), de 3 elementos de entrada, duas camadas escondidas de 4 e 2 neurônios respectivamente, e 1 neurônio de saída, onde as sinapses novas são consideradas totalmente independentes das suas originais.

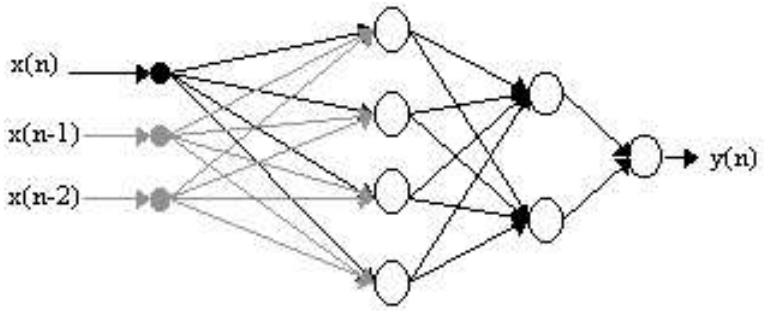

FIGURA 3.4 – Método de janelamento para processamento temporal.

Outra rede bastante conhecida que lida com variações temporais é a TDNN (*Time Delay Neural Networks*), que proporciona memória a todos os neurônios, tanto das camadas ocultas, como da camada de saída. A Figura 3.5 ilustra um exemplo do modelo de rede com atraso no tempo TDNN [82]. A rede mostrada na Figura 3.5 só possui uma camada escondida (podendo ter mais de uma) com memória de ordem 1, e a camada de saída tem memória de ordem 2. De maneira similar ao método de janelamento, existem sinapses novas, mas a rede final formada não conduz à forma da rede estática padrão totalmente interconectada, pois os neurônios novos na camada escondida não estão conectados aos elementos de entrada. Este fato é que faz a rede TDNN perder certa simetria no processo da exploração *feedforward* e de retropropagação da rede, e portanto a torna mais complexa.

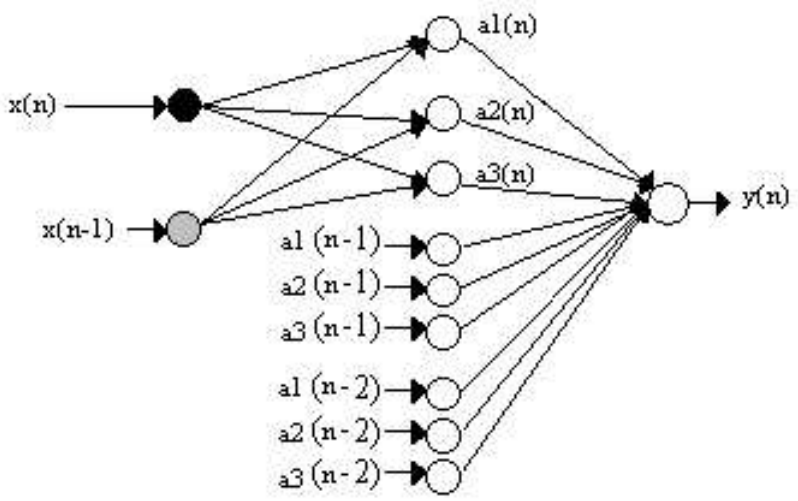

FIGURA 3.5 – Rede TDNN para processamento temporal.

3.3 Redes Neurais Recorrentes

Além da classe de Redes Neurais *feedforward*, apresentada anteriormente, existem as RNAs que possuem arquitetura recorrente. Nesta seção são apresentadas algumas das arquiteturas de RNRs encontradas na literatura, e uma breve descrição sobre algoritmos de aprendizagem utilizados para este tipo de rede. As Redes Neurais Recorrentes (RNRs) são redes que possuem uma ou mais conexões de realimentação (Veja a Figura 3.6). A realimentação pode ser de natureza local se está dada a nível de um neurônio ou de natureza global se a realimentação engloba uma ou mais camadas completas. Quando o *perceptron* de múltiplas camadas tem duas ou mais camadas ocultas, as formas possíveis de realimentação global se expandem ainda mais, sendo assim as RNRs possuem um grande número de arquiteturas.

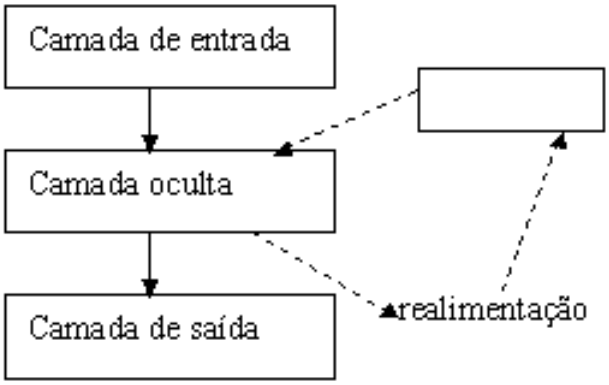

FIGURA 3.6 – Processamento temporal utilizando rede recorrente.

Para as Redes Neurais com atrasos no tempo é necessário, primeiramente, determinar um nível de atrasos ótimo, que na prática é o número de observações prévias a serem usadas como memória. Como é difícil determinar o valor ótimo dos atrasos para um número de observações, é vantajoso que este nível de atrasos esteja incorporado de alguma maneira dentro da rede, característica conhecida como memória flexível de entradas prévias e internas da rede. Esta memória flexível pode ser obtida tornando a rede recorrente, isto é, realimentando respostas geradas pelos neurônios da rede, para serem utilizadas em iterações futuras. Esta realimentação de respostas armazena, indiretamente, dentro da memória, todos os valores prévios apresentados à rede.

Basicamente, são duas as maneiras que as RNRs podem ser utilizadas: como memórias associativas, como por exemplo a rede de Hopfield [39] e como redes para mapeamento de entrada-saída. O espaço de entrada de uma rede de mapeamento é, por definição, mapeado em um espaço de saída. Para este tipo de utilização, uma RNR responde temporariamente a um sinal de entrada aplicado externamente. O uso de realimentação global tem o potencial de reduzir significativamente as exigências de memória. Além disso, a aplicação de realimentação permite que as Redes Recorrentes adquiram representações do estado, o que as torna dispositivos apropriados para aplicações tão diversas como previsão não-linear e modelagem, equalização adaptativa de canais de comunicação, processamento de voz, controle de instalações industriais e diagnóstico de motores automotivos [34].

3.3.1 Redes Recorrentes para Mapeamento Entrada-Saída

As RNRs podem assumir diferentes tipos de arquiteturas, que incorporam um *perceptron* de múltiplas camadas estático ou partes dele, e todas exploram sua capacidade de mapeamento não-linear. Como o objetivo deste trabalho é o estudo do grupo de RNRs usadas para mapeamento de um espaço de entrada em um espaço de saída, algumas das arquiteturas existentes para estas RNRs são apresentadas a seguir, cada uma realçando uma forma de realimentação global.

A arquitetura mostrada na Figura 3.7, representa uma Rede Recorrente genérica que resulta de um MLP. No modelo existe uma entrada que é aplicada a uma memória de linha de atraso derivada com q unidades. A saída é realimentada para a entrada através de uma outra memória de linha de atraso derivada, também com q unidades. Os conteúdos destas duas memórias são utilizados para alimentar a camada de entrada do *perceptron* de múltiplas camadas. A saída, *y(n+1)*, está adiantada em relação à entrada, *u(n)*, por uma unidade de tempo. Assim o vetor de sinal aplicado à camada de entrada do perceptron de múltiplas camadas consiste de uma janela de dados constituída dos valores presente e passados da entrada, *u(n)*, *u(n-1)*, ..., *u(n-q+1)*, que representam entradas exógenas originárias de fora da rede, e dos valores atrasados da saída, *y(n)*, *y(n-1)*, ..., *y(n-q+1)*, sobre os quais é feita a regressão da saída do modelo *y(n+1)*.

A RNR da Figura 3.7, é denominada como um modelo auto-regressivo não-linear com entradas exógenas (NARX, *nonlinear autoregressive with exogenous inputs*), sendo seu comportamento dinâmico descrito por

$$y(n+1) = F(y(n), ..., y(n-q+1), u(n), ..., u(n-q+1)) \tag{3.1}$$

onde F é uma função não-linear de seus argumentos.

Neste tipo de arquitetura a entrada é realimentada com os valores atrasados da saída, juntamente com o valor atual da variável de entrada e seus valores atrasados no tempo. Como pode ser visualizado na Figura 3.8, a memória é aplicada como entrada para o processamento da rede, mas existe também um laço de realimentação da saída para a memória, já que são usados também, valores da saída atrasados no tempo.

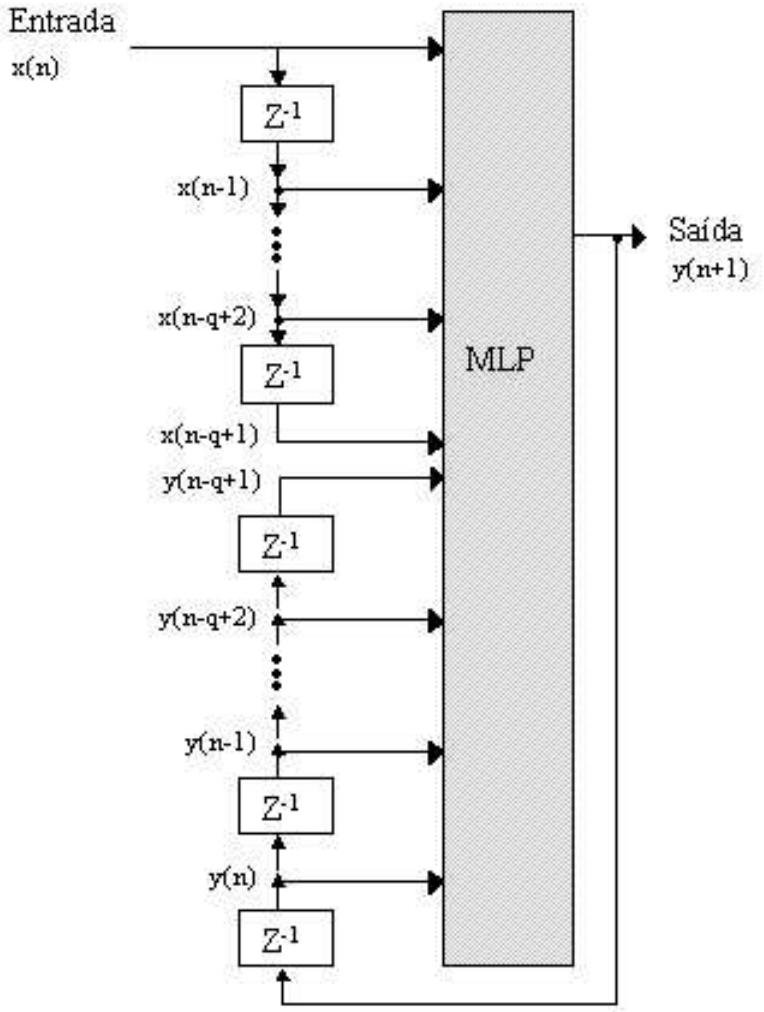

FIGURA 3.7 – Modelo Auto-Regressivo Não-Linear com Entradas Exógenas (NARX) [34].

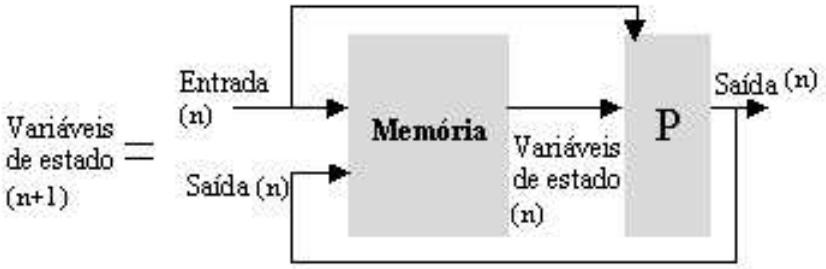

FIGURA 3.8 – Representação das características de uma Rede NARX, através de um modelo de circuito seqüêncial.

Um outro tipo de arquitetura é o modelo de espaço de estados, onde os neurônios ocultos definem o estado da rede, e a saída da camada oculta é realimentada para a camada de entrada através de um banco de atrasos unitários. A camada de entrada consiste de uma concatenação de neurônios realimentados e neurônios de entrada, que conectam a rede ao ambiente externo. O número de atrasos unitários usados para realimentar a saída da camada oculta de volta para a camada de entrada, determina a ordem do modelo.

Outros dois exemplos de RNRs são a Rede de Elman [23] e a Rede de Jordan [42]. A Rede de Elman, mostrada na Figura 3.9, contém conexões recorrentes dos neurônios ocultos para uma camada de unidades de contexto que consiste de atrasos unitários. As unidades de contexto armazenam as saídas dos neurônios ocultos por um passo de tempo, e então as realimentam de volta para a camada de entrada, podendo ser consideradas como atraso no tempo em um passo. Os neurônios ocultos têm assim um registro das suas ativações passadas, o que dá capacidade para a rede realizar tarefas de aprendizagem que se estendem no tempo.

A Rede de Elman utiliza um laço de realimentação para a camada oculta, o que propicia a memória de suas ativações por um passo de tempo. Assim, de acordo com a Figura 3.10, a memória armazena o resultado do processamento dos valores das variáveis de estado e de entrada, direcionando-o para o processamento de saída e realimentando-o no processamento de entrada.

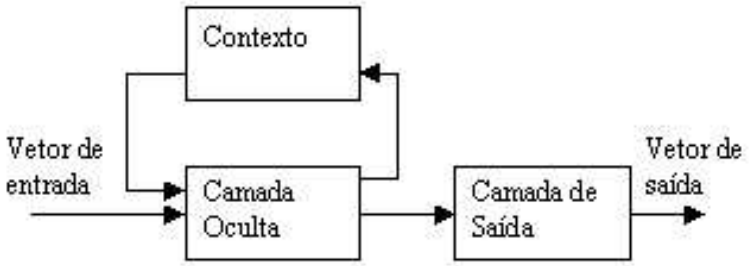

FIGURA 3.9 – Arquitetura de uma Rede de Elman.

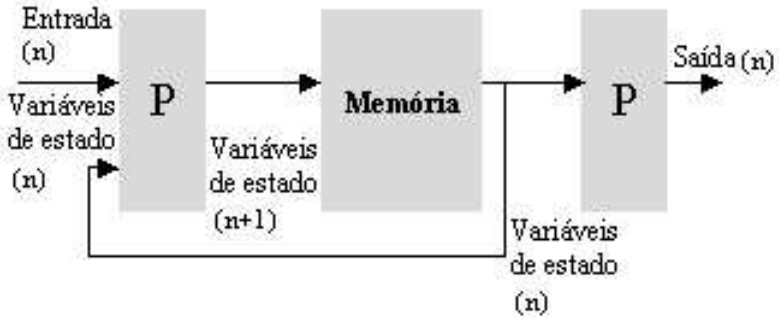

FIGURA 3.10 – Representação das características de uma Rede de Elman, através de um modelo de circuito seqüêncial.

Na Rede de Jordan, a saída é copiada para a unidade de contexto. Além disso, as unidades de contexto são localmente recorrentes. A grande diferença em termos de topologia entre as duas redes é que a recorrência na rede de Elman é feita da camada oculta para as entradas, enquanto na rede de Jordan a recorrência é feita das saídas para as entradas, conforme mostra a Figura 3.11.

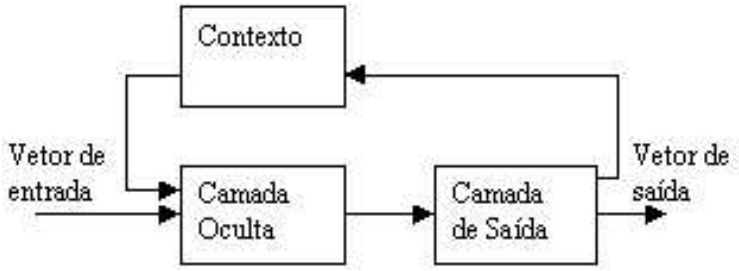

FIGURA 3.11 – Arquitetura de uma Rede de Jordan.

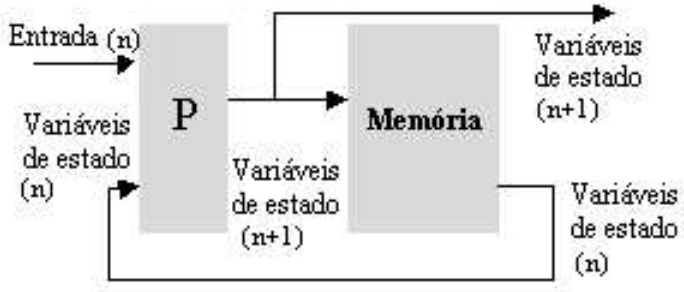

FIGURA 3.12 – Representação das características de uma Rede de Jordan, através de um modelo de circuito seqüêncial.

Neste modelo, as variáveis de entrada e de estado são aplicadas para o processamento de entrada, sendo os valores destas variáveis de estado armazenados na memória, para serem realimentados como entrada, e também, direcionados como saída, como pode ser visto na Figura 3.12. Desta forma a memória não está localizada como fonte de entrada para a rede, mas ela é obtida através da rede e realimentada como entrada.

3.3.2 Algoritmos de Aprendizagem

Duas formas de treinamento que podem ser usadas para uma RNR, e que envolvem o uso de aproximações na computação dos gradientes, são retropropagação através do tempo (BPTT, *Back-Propagation Through Time*) e Redes Recorrentes de tempo real (RTRN, *Real-Time Recurrent Networks*).

O algoritmo BPTT para o treinamento de RNRs é uma extensão do algoritmo *back-propagation* padrão que efetua a operação temporal em um rede MLP onde a topologia da rede é acrescida de uma camada para cada instante de tempo.

No método utilizado pelo algoritmo BPTT, a rede é expandida no tempo. Ao final da seqüência de entrada, os valores esperados são apresentados, e o gradiente é calculado retropropagando o sinal do erro no tempo. Este método possui a desvantagem de que nenhum aprendizado é efetuado até que se alcance o fim da seqüência.

Outros modelos recorrentes treinados para funcionar continuamente são as RTRNs [88]. Uma característica particular destas redes reside em sua habilidade para lidar tanto com as entradas quanto com as saídas variando no tempo, através do seu próprio funcionamento temporal.

A capacidade das RTRNs de proporcionar uma dinâmica arbitrária torna estas redes uma ferramenta útil em aplicações de tempo real, como modelagem biológica [3], aplicações de lingüística [29] e aplicações de fala [34].

3.4 Extração de Conhecimento a partir de Redes Neurais Temporais

Nesta seção é apresentada a extração de conhecimento a partir de Redes Neurais Temporais, sendo citados alguns trabalhos envolvendo a extração de conhecimento e alguns tipos de formalismos utilizados para a sua representação. O objetivo da extração de conhecimento é gerar uma descrição simbólica concisa do conhecimento armazenado nos pesos de uma RNA [61], [48], [4]. Através da extração de conhecimento, podem ser descobertas características que não são identificadas previamente e relacionamentos não lineares em conjuntos de dados, que possibilitam o aumento de desempenho e generalização, e ainda, a sua utilização em outros problemas de aprendizagem semelhantes.

Como aproximadores universais, uma vez que se disponha de um conjunto de dados, as RNAs podem ser usadas para estimar, com um certo erro, o comportamento do processo que gerou os dados sob as mesmas condições usadas para gerá-los. Assim, a rede passa a representar um modelo limitado do processo, que armazena, após o treinamento, as características dos dados gerados pelo processo.

Como as RNAs implementam um mapeamento aprendido a partir de dados, alguns compor-

tamentos não desejados podem ser introduzidos neste processo. Por exemplo, o comportamento da Rede Neural fora da região dos dados de treinamento depende da arquitetura da Rede Neural e do processo de treinamento. Este comportamento pode ser analisado através da extração de conhecimento. A representação correspondente à Rede Neural em um paradigma simbólico é geralmente mais simples de entender [60].

A validação da RNA é geralmente representada por sua capacidade de generalização. A generalização refere-se à capacidade da Rede Neural de produzir resultados adequados dentro da região do espaço de entrada que contém os dados, sendo que esta região pode ser definida como o menor conjunto convexo que contenha os dados e, à capacidade da Rede Neural de produzir resultados adequados fora desta região. A análise da função da RNA (extração do conhecimento) pode facilitar esta tarefa.

Esta validação das RNAs pode ser realizada pontualmente através do erro de aproximação, que fornece uma validação estatística da rede através de uma amostragem da função realizada pela rede. A dificuldade desta abordagem está na dependência do número de amostras necessárias para realizar a amostragem, com o número de entradas da Rede Neural.

O problema da extração de conhecimento a partir de RNRs possui relação com o aprendizado e análise de séries temporais. Este é dividido em: previsão, geração, reconhecimento e segmentação de seqüências. As Redes Recorrentes podem ser utilizadas para tratar os três primeiros subproblemas. A segmentação, por outro lado, é realizada normalmente através de algoritmos de "clusterização".

Toda extração de conhecimento pressupõe uma forma de representação. Uma das possibilidades de expressar o conhecimento adquirido pela Rede Neural em uma forma compacta e de fácil entendimento é com a utilização de autômatos finitos [81]. Uma grande quantidade de trabalhos tem apresentado o aprendizado, síntese e extração de autômatos finitos em RNRs, assim como a utilização de Redes Neurais treinadas para se comportar como um autômato finito difuso. Alguns destes trabalhos podem ser encontrados em [8], [30], [29].

Existem diversas formas de representar o conhecimento armazenado nas Redes Neurais Temporais, entre elas estão o autômato de estados finito determinístico, o autômato de estados finito difuso, as equações diferenciais e os modelos de Markov [7]. Em relação ao problema da extração de conhecimentos a partir de RNRs treinadas com seqüências temporais simbólicas, pode ser demonstrado que um autômato de estados finito determinístico completo e suas classes equivalentes podem ser extraídas a partir de RNRs [29], [40], [14], embora nenhum dos trabalhos apresente um método para tal extração. A ativação dos neurônios nas camadas escondidas representa o passado e grupos destas ativações representam estados do autômato gerado. O conhecimento pode também ser representado através de equações diferenciais. [16] e [63] apresentam metodologias para realizar a transformação de equações diferenciais em uma arquitetura especial de Redes Neurais (*Higher-Order* RBF ou *Higher-Order Modified Logistic Networks*).

Autômatos Finitos Determinísticos podem ser sintetizados por ou mapeados em uma RNR diretamente pela programação da estrutura do autômato nos pesos da Rede Neural. Quando visualizadas como autômato, RNRs podem ser descritas em termos de máquinas de estados [9]. Ativações de unidades de estado representam a história passada e *clusters* dessas ativações podem representar os estados do autômato gerado [80].

Quanto à extração de modelos locais de equações diferenciais, há dois tipos: linear e não-linear. Os coeficientes de um conjunto de equações lineares podem ser obtidos através da análise de sensibilidade da Rede Neural ou a partir de metodologias, como apresentado em [10] (válidos em estados difusos) ou [54] e [55] (válidos em estados definidos por polígonos). Para modelos locais não-lineares com base conhecida, uma extensão daqueles métodos se faz necessária. Outros métodos são também discutidos em [20], [81], [61].

Capítulo 4

Extração de Conhecimento a partir de Redes Neurais Recorrentes

4.1 Introdução

No presente trabalho propõe-se a extração de conhecimento utilizando uma clusterização difusa, que possibilita a representação do conhecimento através de formalismos simbólicos. Este tipo de clusterização difusa é comparado com a clusterização através do algoritmo *K-means*. São estudadas e analisadas, basicamente, três formas para extração do conhecimento: Autômatos Finitos Determinísticos, Autômatos Finitos Difusos e Cadeias de Markov. Comparou-se o poder representativo nestes três formatos segundo aspectos descritos em [4] e [17]: compreensibilidade, fidelidade, exatidão e generalidade. O conhecimento deve ser extraído de forma a facilitar o entendimento do sistema aprendido pela Rede Neural, sendo que a resposta fornecida pela rede deve ser a mais idêntica possível à resposta do sistema extraído. Segundo [17], os modelos mais precisos tendem a ser incompreensíveis e portanto ele propõe a abordagem do relaxamento da fidelidade para solucionar o problema da compreensibilidade, sendo esta abordagem também usada para este trabalho. A possibilidade de obter uma representação com alto grau de fidelidade, acarretaria em um modelo de difícil visualização e compreensão, sendo assim o que se obtém, é um modelo que respeite as características apresentas pelos sistemas estudados, mas que também possibilite a compreensão através da visualização do modelo simbólico obtido.

Para realizar uma comparação entre o formalismo extraído e a Rede Neural Temporal, deve-se levar em consideração o tipo de Rede Neural que foi empregado para armazenar o conhecimento, o método usado para realizar a extração e o formato da informação que se pretende extrair [79].

Os aspectos citados acima são utilizados como referência para a validação das Redes Neurais treinadas e também para os formatos simbólicos extraídos. A validação é efetuada através de simulação e análise gráfica dos diversos modelos obtidos, comparando-se cada um deles com o modelo de equações dos sistemas utilizados como aplicação.

As aplicações tratadas no trabalho, são séries temporais calculadas através de simulações da dinâmica dos sistemas abordados. Para o sistema do Pêndulo Inverso, a seqüência de posições

que o pêndulo assume ao longo de possíveis trajetórias são utilizadas para o treinamento de uma RNR, e para o sistema de Lorenz, são utilizadas séries temporais geradas a partir das três equações do sistema, cobrindo, da melhor forma possível, o seu espaço de estados. Estes sistemas dinâmicos não lineares serão especificados no capítulo seguinte.

Os métodos para extração do conhecimento e os três tipos de formalismos simbólicos utilizados para a representação deste conhecimento são apresentados no decorrer do capítulo, salientando suas características e principais diferenças observadas através de comparação entre os formatos.

Para o melhor entendimento dos formalismos apresentados é necessário definir alguns conceitos adotados para os métodos de extração de conhecimento. Em um espaço de estados, um determinado estado é um ponto localizado dentro de um mapa, sendo que para um autômato ou uma cadeia de Markov, um estado é representado através de uma região do espaço de estados, usando como referência um centróide, que é o ponto médio desta região, podendo também, ser considerado como um conjunto difuso quando se refere a um autômato difuso.

Um Autômato Finito Difuso (AFDif) [13], é uma generalização difusa de autômato finito determinístico. Na representação difusa, o estado corrente de um autômato é uma coleção de estados de um AFDet, sendo estes estados ocupados com diferentes graus de pertinência. Redes Neurais tem sido treinadas para se comportar como AFDifs não sendo necessário carregar esta propriedade, sua representação interna de estados e transições podem se tornar instáveis para seqüências de entrada suficientemente longas [80].

Em contraste com o AFDet, um conjunto de estados do AFDif pode ser ocupado com variação de graus em qualquer ponto do tempo, geralmente diminuindo o tamanho do modelo, e tornando o sistema dinâmico inicialmente modelado mais acessível para a interpretação.

Para as extrações de conhecimento, usadas no trabalho, atribui-se significado à uma região do espaço de atividades dos neurônios. Associa-se os estados contínuos pelos quais uma Rede Neural Temporal passa à medida que esta é simulada à estados discretos, estados difusos ou graus de pertinência à modelos locais de equações diferenciais.

4.2 Extração de Autômatos Finitos Determinísticos a partir de Redes Neurais Recorrentes

Esta seção descreve a metodologia adotada para a extração de autômatos finitos determinísticos. Neste caso, basicamente, a extração dos estados discretos do autômato dá-se pela atribuição de uma combinação de intervalos na atividade conjunta dos neurônios a um estado. Esta atribuição envolve critérios de "clusterização", que dependem da fidelidade do autômato desejado (e portanto da complexidade do autômato extraído).

O método mais comumente utilizado para clusterização, é o *K-means*, que tem como objetivo particionar um conjunto de dados D com n observações em k grupos distintos. Através da escolha de k valores distintos para a escolha dos centros dos grupos, podendo ser esta uma escolha aleatória, define-se os valores dos centróides de cada *cluster*. Associando-se cada ponto amostrado ao centro do grupo, e recalculando o centróide de cada grupo, obtêm-se em cada

cluster, os pontos amostrados com maior similaridade. Para determinar a que cluster cada ponto pertence pode ser aplicada a métrica de similaridade dada pela Distância Euclidiana [49]. A Figura 4.1 apresenta uma visualização desta abordagem.

Este método de clusterização é utilizado no trabalho, tendo em vista a possibilidade de escolha do número de estados que se deseja representar em um determinado autômato. Sendo a escolha do número de estados que melhor represente um sistema, efetuada através da análise de diversos experimentos executados para a aplicação, variando-se o número de estados utilizados para a representação do AFDet.

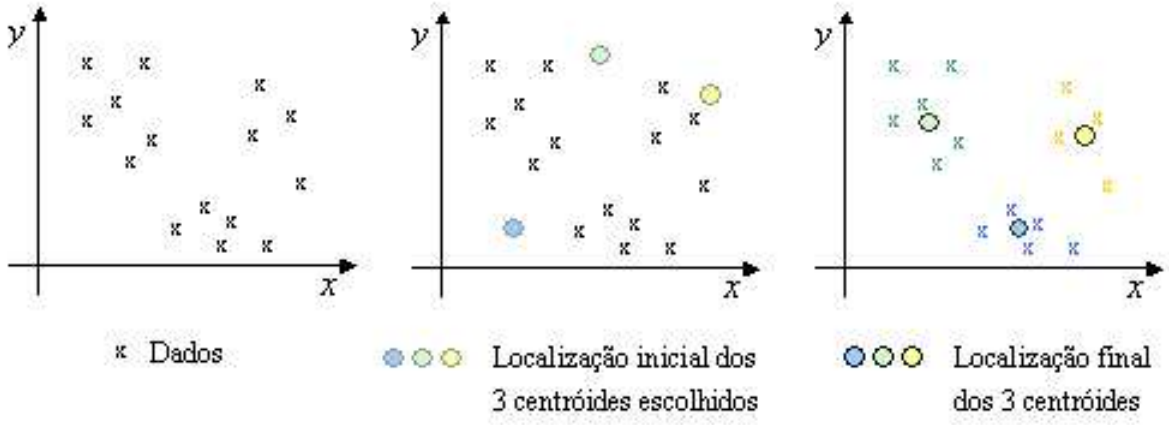

FIGURA 4.1 – Clusterização utilizando o método *K-means*. O primeiro gráfico (esquerda) apresenta uma distribuição de dados amostrados em um espaço 2D, o segundo gráfico (central) mostra três centróides para os três clusters escolhidos, dispostos aleatoriamente no espaço e no último gráfico (direita) são mostrados os três *clusters* devidamente identificados, com os centróides deslocados para a posição central de acordo com o seu *cluster*.

Uma dificuldade na interpretação do conhecimento contido nas Redes Neurais como um autômato determinístico, está na discretização do espaço de estados em um número finito de regiões.

A investigação do método mais adequado para realizar a conversão dos estados difusos em estados clássicos do autômato sem perda considerável de informação, pode ser feita através de três abordagens: considerando o estado como tendo um raio constante, considerando o estado como tendo um raio variável e utilizando a definição de diagramas de Voronoi. Através da utilização de um raio constante se determina um tamanho fixo a ser aplicado a cada estado, ou um ponto que represente cada um dos estados. Para um raio variável é necessário determinar o número de funções de pertinência que melhor representam as ativações de cada neurônio oculto, variando de um neurônio para outro. Utilizando a definição de diagramas de Voronoi, considera-se que os estados representem a região delimitada pelo diagrama, que considera como referência o centróide para os dados amostrados.

O Diagrama de Voronoi de um conjunto finito S de n pontos em um plano é uma partição do plano em n regiões de modo que cada região i da partição, $i = 1, ..., n$, é o lugar geométrico de

pontos que estão mais próximos do i-ésimo membro de S do que de qualquer outro membro [46]. Um exemplo do diagrama de Voronoi pode ser observado na Figura 4.2, que mostra inicialmente os pontos em um plano e após, a estrutura geométrica das regiões definidas para cada ponto, formando o diagrama.

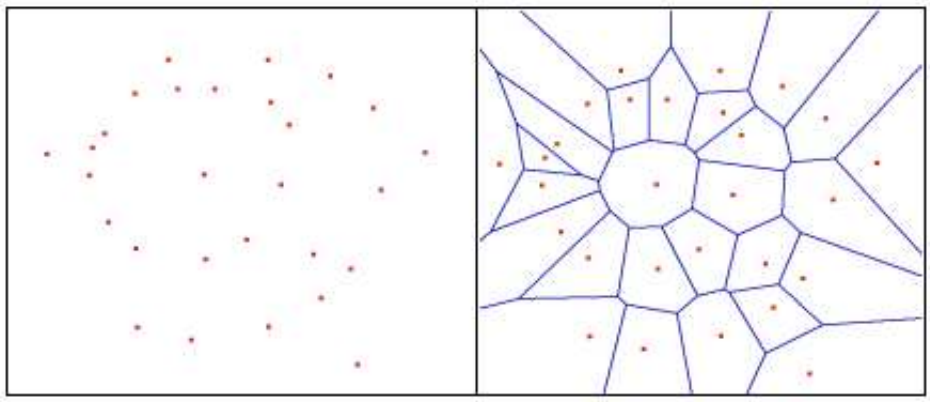

FIGURA 4.2 – Exemplo de um diagrama de Voronoi.

A fim de obter uma visualização do AFDet como um diagrama de Voronoi, utiliza-se os centróides de cada *cluster*, como ponto de referência para a construção do diagrama. Desta forma, cada região mostrada no diagrama caracteriza os estados referentes a cada centróide. A Figura 4.3 mostra a representação de um AFDet na forma de um diagrama de transições e, ao lado, na forma de um diagrama de Voronoi, onde cada região delimita um estado que é representado pelo seu centróide. É possível visualizar também, as possíveis transições entre os três estados do autômato, sendo que no AFDet as transições ocorrem de forma discreta entre os estados.

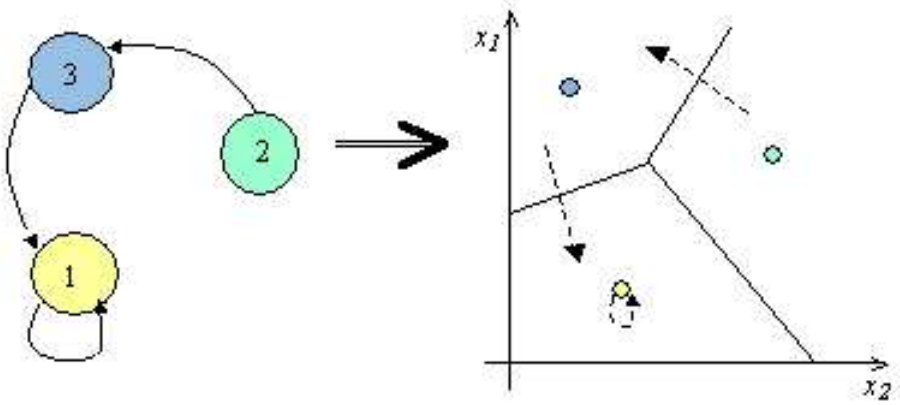

FIGURA 4.3 – Representação de um AFDet com três estados na forma de diagrama de transições e de um diagrama de Voronoi.

Como os estados são representados através de regiões no espaço de estados, como pode ser visto no diagrama de Voronoi da Figura 4.3, a identificação das transições é efetuada através das séria temporais geradas para o sistema, sendo uma transição detectada quando o sistema estiver em um ponto limite do seu estado atual e passar para um ponto dentro da região delimitada para um estado próximo a ele.

4.3 Extração de Cadeias de Markov a partir de Redes Neurais Recorrentes

Nesta seção, é apresentada a metodologia utilizada para a extração de cadeias de Markov, sendo esta idêntica a extração de AFDet, já que, para este tipo de extração, também torna-se necessário determinar o número de estados que se deseja representar, se o método adotado para a clusterização for o *K-means*. Se o método de clusterização escolhido for através da utilização de estados difusos, deve-se trabalhar com o número de estados encontrados, ou com os estados que representem a maioria dos dados amostrados.

A Figura 4.4 mostra uma cadeia de Markov com três estados, na forma de um diagrama de transições e de um diagrama de Voronoi. Observa-se a possibilidade de, estando o sistema em um determinado estado, escolher entre outros dois estados para transicionar, característica que, além da associação de probabilidades às transições, o diferencia de um AFDet. Neste formalismo, tanto a duração do sistema dentro de um estado, como o estado para onde o sistema irá passar, são determinados através de probabilidades determinadas através das características apresentadas pelo sistema.

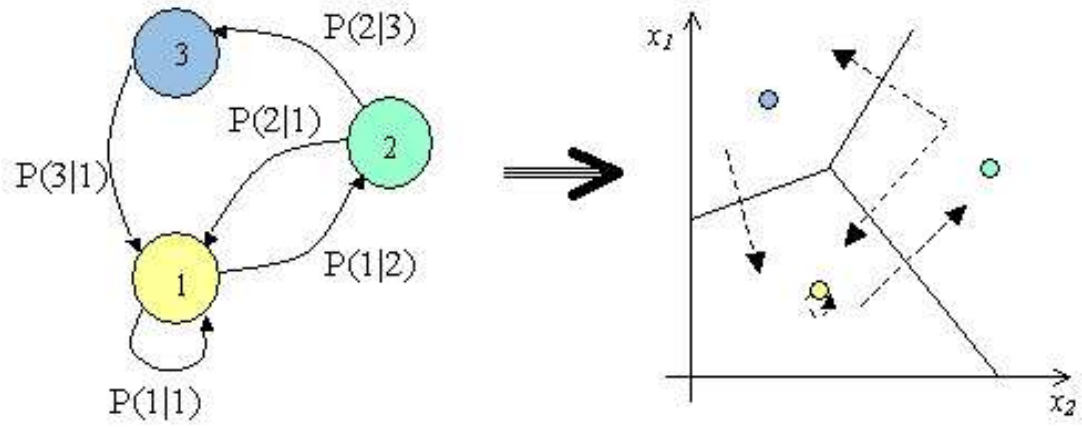

FIGURA 4.4 – Representação de uma Cadeia de Markov com três estados na forma de diagrama de transições e de um diagrama de Voronoi.

Para a extração de cadeias de Markov, é utilizado o mesmo processo adotado para a extração de autômatos finitos determinísticos, sendo que as probabilidades necessárias aos estados

e transições são observadas a partir de simulações executadas para o sistema, com diferentes parâmetros de inicialização. Utiliza-se a freqüência de transições encontradas de acordo com os dados das simulações, para aproximar os valores de probabilidade de cada transição ocorrer.

4.4 Extração de Autômatos Finitos Difusos a partir de Redes Neurais Recorrentes

Outra forma de representar o conhecimento extraído de uma RNR é através de um autômato finito difuso, sendo, a metodologia para tal, é apresentada no decorrer desta seção. Neste tipo de representação simbólica, um conjunto de estados é ocupado com diferentes graus de pertinência em um dado instante de tempo.

Regras difusas são uma forma de representação do conhecimento armazenado na Rede Neural Temporal, pois: as entradas da Rede Neural podem ser diretamente interpretadas como o estado do sistema; o mapeamento realizado pela rede, do estado do sistema para as saídas pode ser interpretado como um mecanismo de inferência e as saídas da rede podem ser interpretadas como as ações que o sistema deve realizar bem como o próximo estado a ser assumido pela Rede Neural. Exemplos de regras utilizadas na representação do conhecimento podem ser encontradas em [10].

Há basicamente dois tipos de regras difusas: Mamdani e Sugeno [50]. As regras do tipo Mamdani mapeiam conjuntos difusos para conjuntos difusos, e portanto são adequadas para representar transições de estados difusos. Regras do tipo Sugeno [77] mapeiam conjuntos difusos para sistemas de equações entre o espaço de entrada e o espaço de saída da Rede Neural.

As vantagens da solução difusa são: o paralelismo, pois permite um controle complexo por combinação de regras simples podendo a inferência ser realizada em paralelo; o controle lógico, que é especialmente adequado à aquisição de conhecimento a partir de peritos humanos pois permite a definição de regras em linguagem comum; e o controle lingüístico, pois as regras podem ser facilmente compreendidas pelos operadores que podem até interpretar o efeito de cada regra.

Em [61] é apresentado um algoritmo para converter AFDifs em RNRs, porém usando uma arquitetura especial: Redes Neurais Recorrentes de segunda ordem, ou seja, contendo neurônios multiplicativos e uma estrutura especial adequada à extração do AFDif. Blanco et al., em [8], usa também uma arquitetura especial de segunda ordem. Esta é uma imposição do processamento de regras difusas, cujos mecanismos de inferência e operação lógica AND usam normalmente a operação de multiplicação como implementação.

Para a extração de AFDifs, os estados de uma Rede Neural devem ser associados aos graus de pertinência em conjuntos difusos. De acordo com a metodologia explanada em [10] é necessário se dispor da Rede Neural, pois o espaço utilizado para a extração é o espaço neural, ou seja, trabalha-se diretamente com os valores das ativações da camada oculta.

A Rede Neural aprende uma aproximação para o modelo dinâmico pela observação das instâncias das transições de estados, bem como das entradas de controle e/ou forças externas que causam estas transições.

Para o método proposto, o primeiro passo a ser realizado para que o AFDif possa ser extraído é o treinamento da Rede Neural. Através dos erros de treinamento e teste da mesma, são determinadas as possíveis topologias a serem utilizadas para a extração. Deve-se considerar que quanto maior o número de neurônio na camada oculta da rede, maior a complexidade do autômato extraído, e principalmente, maior o custo computacional associado ao processo de extração.

Os dados necessários para o treinamento da Rede Neural devem ser gerados, respeitando os parâmetros fornecidos para o modelo. Várias simulações com as equações do sistema dinâmico devem ser executadas, de forma a tornar possível a extração de todos os comportamentos possivelmente encontrados no sistema, para isso, foram desenvolvidas implementações para a geração dos dados e simulação das aplicações estudadas.

A ferramenta utilizada para o treinamento das Redes Neurais é o programa *SNNS* (*Stuttgart Neural Network Simulator*) [75], um simulador de RNAs criado na Universidade de Stuttgart, que proporciona um ambiente eficiente e flexível para a criação, treinamento e manutenção de Redes Neurais Recorrentes ou não. Através da Rede Neural implementada em C, fornecida pelo SNNS, pode-se realizar a integração da rede com as demais implementações para a extração de conhecimento.

Para a aplicação do método de clusterização difusa, é necessário, após o treinamento e teste da Rede Neural, apresentar para a mesma os dados gerados nas simulações, possibilitando a análise das ativações encontradas nos neurônios da camada oculta. Através da clusterização no espaço de ativações dos neurônios na camada oculta, são construídas as funções de pertinência adequadas a cada neurônio da camada oculta.

Duas metodologias podem ser utilizadas para a identificação do número de funções de pertinência aplicadas a cada neurônio. Uma metodologia que pode ser utilizada, é a construção de três funções de pertinência para cada um dos neurônios da camada oculta, possibilitando que o método seja aplicado para Redes Neurais com qualquer número de neurônios ocultos, tornando-o independente de análises prévias das funções de ativação dos neurônios. Porém o número de estados encontrados através da combinação destas funções tende a aumentar drasticamente, de acordo com o número de neurônios utilizados, já que, é necessário combinar todas as funções de todos os neurônios. Neste caso, pode-se aproximar a função de ativação de cada neurônio da camada oculta por três equações lineares idênticas e fixas para todos os neurônios. As funções $F1(x)$ (4.1), $F2(x)$ (4.2) e $F3(x)$ (4.3), apresentadas na Figura 4.5, representam as três funções de pertinência utilizadas para cada neurônio da camada oculta.

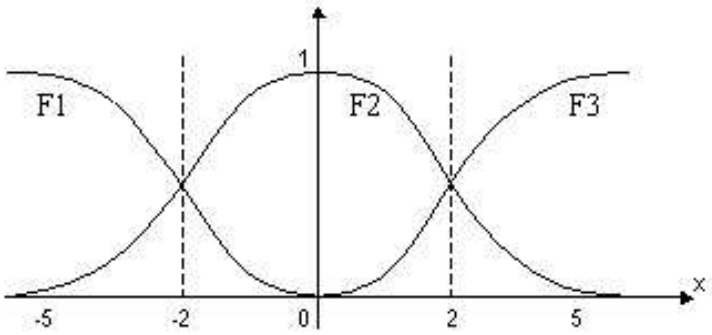

FIGURA 4.5 – Funções de pertinência ideais para cada região do neurônio com função sigmóide.

$$F1(x) = \begin{cases} -sigm(x) + sigm'(x)x & ,x < 0 \\ 0 & ,x \geq 0 \end{cases} \tag{4.1}$$

$$F2(x) = 2 * sigm'(x) \tag{4.2}$$

$$F3(x) = \begin{cases} sigm(x) - sigm'(x)x & ,x > 0) \\ 0 & ,x \leq 0 \end{cases} \tag{4.3}$$

onde:

$$sigm(x) = f(x) = \frac{1}{1 + e^{-x}} \tag{4.4}$$

$$sigm'(x) = f'(x) = f(1 - f) \tag{4.5}$$

A idéia principal da segunda metodologia é usar somente uma parte do espaço neural para computar os conjuntos difusos, ou seja, para construir as funções de pertinência. A conseqüência está em obter um número variável de funções de pertinência para cada neurônio oculto. Estas funções são válidas somente em seu sub-espaço do espaço neural. Isto permite a melhor correspondência entre as funções de pertinência e a rede no sub-espaço. O objetivo desta metodologia também é reduzir o número de funções de pertinência generalizadas, assim, reduzindo o número de estados do autômato.

A construção das funções de pertinência é baseada na função Gaussiana, considerando-se a média e o desvio-padrão dos valores de ativação encontrados nas regiões de trabalho de cada neurônio da camada oculta. Para neurônios que trabalham somente em uma região linear da função de ativação, como pode ser observado nos gráficos à esquerda da Figura 4.6, uma única função de pertinência pode ser construída, ou seja, um único conjunto difuso é encontrado de

acordo com a região de trabalho da função de ativação do neurônio. Para neurônios que trabalham ao longo da função de ativação, ou seja, em várias regiões lineares da função de ativação, como mostram os gráficos à direita, é necessária a construção de uma função de pertinência adequada a cada uma destas regiões.

Através da combinação das funções de pertinência de todos os neurônios, são construídas as funções de pertinência (*G's*) associadas a cada estado. Por exemplo, para a Figura 4.6, obter-se-iam 3 funções de pertinência através da combinação das funções de pertinência *F1* x *F1'*, *F1* x *F2'* e *F1* x *F3'*. Estas combinações são implementadas através do produto, ou AND na lógica difusa. Neste tipo de clusterização, chamada difusa, o centróide associado a cada estado, ou seja, a cada função de pertinência (*G's*), é o dado que apresenta o maior grau de pertinência no estado.

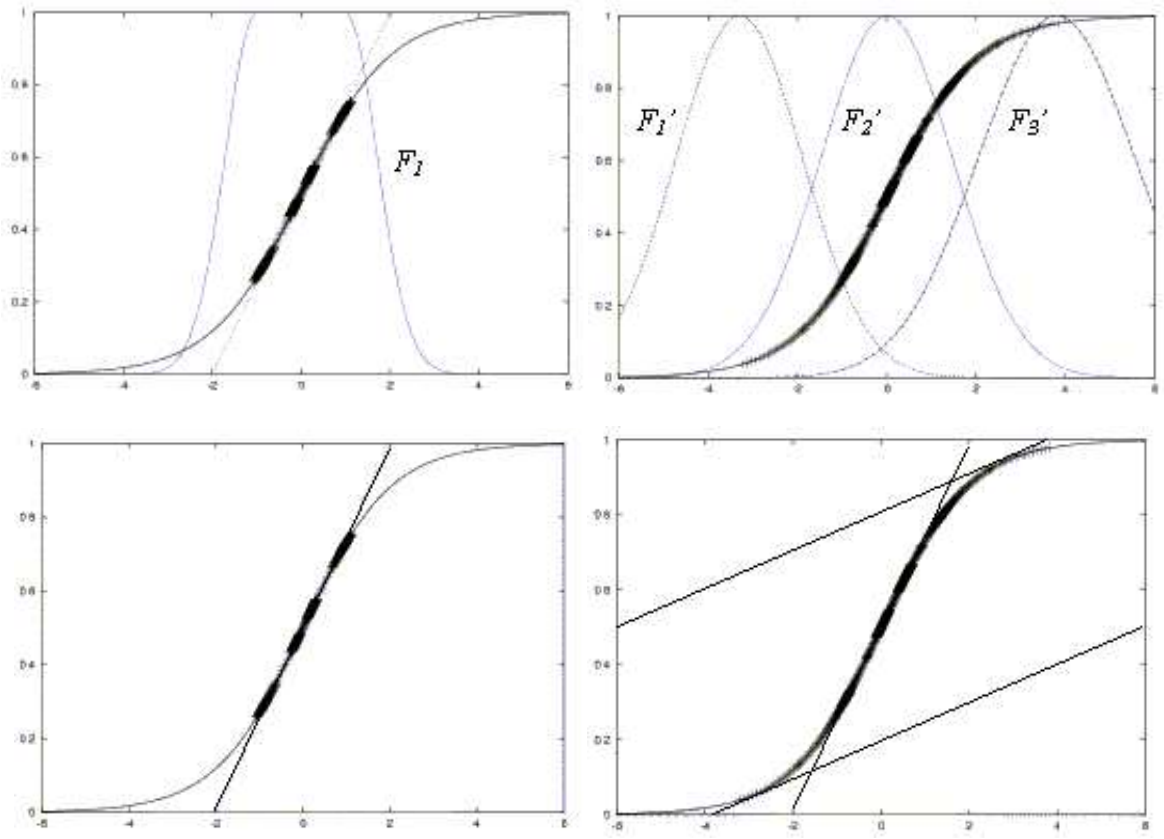

FIGURA 4.6 – Ativações para dois neurônios na camada escondida durante simulações. Observa-se as regiões de trabalho da função de ativação dos neurônios não-lineares. Neurônio que trabalha somente na região linear central da função de ativação: 1 função de pertinência (esquerda). Neurônio que trabalha em 3 possíveis regiões lineares: central, esquerda e direita: 3 funções de pertinência (direita).

Um esquema básico contendo 6 passos que descrevem a aplicação do método de extração de um AFDif utilizando a clusterização difusa é apresentado no quadro da Figura 4.7

1 ⇒	Obter os valores de ativação da camada oculta para os dados de treinamento
	→ Apresentar dados de entrada para rede do treinamento
	→ Armazenar valores de ativação para o cálculo das *F's*
2 ⇒	Calcular as funções de pertinência para cada neurônio (*F's*)
	→ Armazenar valores das *F's* para o cálculo das *G's*
3 ⇒	Calcular as funções de pertinência para cada estado (G's)
	→ Armazenar valores das *G's*
4 ⇒	Identificar quais os estados utilizados que representam os dados
5 ⇒	Identificar as transições através das seqüências temporais
	→ Repetir os passos 1, 2, 3 e 4 para os dados de saída do treinamento
	→ Comparar os valores das *G's* dos dados de entrada e de saída
	→ Identificar qual estado representa cada um dos dados
	→ Identificar, para os dados de entrada e de saída, as transições de estado
6 ⇒	Identificar os valores referentes aos centróides nas ativações
	→ Representar os centróides através das variáveis de entrada

FIGURA 4.7 – Esquema básico para a extração de um AFDif.

Neste tipo de autômato, os estados estão associados a conjuntos difusos, onde existem graus de pertinência que indicam em que estado o sistema se encontra com maior pertinência, determinando assim o seu estado atual.

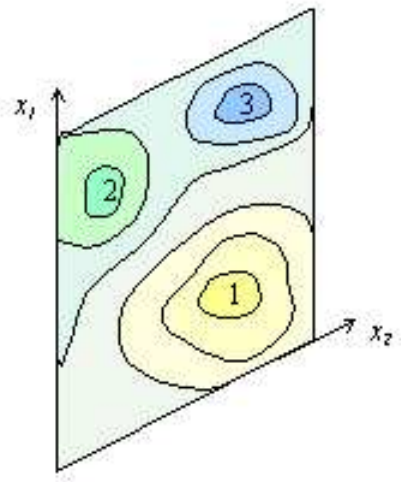

FIGURA 4.8 – Representação gráfica de um AFDif com três estados, representados pelas cores amarelo, verde e azul.

A transição entre os estados difusos caracteriza-se pelo aumento no grau de pertinência do sistema em relação a um outro estado, sendo assim, a transição é caracterizada quando um outro estado possuir um grau de pertinência mais elevado do que o estado atual. Na Figura 4.8 pode ser

observado, através da utilização de cores para a representação dos estados de um AFDif, as regiões de cada estado de acordo com a sua função de pertinência. Os tons mais acentuados representam as regiões onde os estados são mais pertinentes, e as regiões que apresentam decremento em relação a estes tons, são aquelas onde o grau de pertinência de um estado está diminuindo e os graus dos demais estados está se elevando.

Uma forma gráfica de visualização para o AFDif é através da representação de um autômato clássico, através de círculos e setas, indicando os estados e as transições. Esta visualização será realizada através do pacote *FSA* 6.2 (*Finite State Automata Utilities*) [59]. Para conseguir uma visualização mais clara, um modelo bastante apropriado é um gráfico de 3 dimensões, contendo as curvas produzidas através das funções de pertinência de cada estado, sendo que através destas pode-se visualizar as possíveis transições entre os estados encontrados.

Capítulo 5

Aplicações e Resultados

5.1 Introdução

Com o objetivo de extrair uma representação formal do conhecimento armazenado em Redes Neurais Temporais, especificamente em Redes Neurais Recorrentes, selecionou-se duas aplicações que apresentam o comportamento de sistemas dinâmicos não lineares.

A primeira seção apresenta a aplicação do Sistema do Pêndulo Inverso, que caracteriza-se pela não linearidade e instabilidade de seu comportamento, e que tem sido tratada por diferentes métodos, servindo como um bom parâmetro para comparação [5]. Outra aplicação escolhida e apresentada na segunda seção, é o Sistema de Lorenz, implementado através de três equações diferenciais, que representam um sistema dinâmico caótico [62]. As seções apresentam, além da descrição dos sistemas, os experimentos e resultados obtidos para cada aplicação.

5.2 O Pêndulo Inverso

O Sistema do Pêndulo Inverso sobre um carro é um problema clássico de controle e seguidamente é usado como ponto de partida para o estudo sobre controladores [65]. O que há de interessante neste sistema, é que não é difícil controlar uma variável de estado, mas o controle de duas variáveis de estado demanda uma certa inteligência do controlador [11]. O objetivo deste trabalho não é controlar o pêndulo, mas analisar o seu comportamento e extrair a sua dinâmica na forma de um autômato.

O carro pode se mover dentro de um determinado intervalo, sendo que o pêndulo localizado sobre o carro deve permanecer na posição vertical. A posição x do carro é delimitada pelo intervalo -3 $< x <$ 3 $[m]$ e o ângulo ϕ do pêndulo por -$\pi/2 < \phi < \pi/2$. A força F aplicada no carro, provê o sinal de controle, variando continuamente entre -10 $< F <$ 10 $[N]$.

Associadas com o processo do Pêndulo Inverso existem quatro variáveis de estado, que descrevem todas as possíveis configurações que o processo pode assumir. O conhecimento dos valores variáveis de estado permite computar seus valores no futuro, pois o sistema é determinístico. As variáveis de estado, juntas, formam um vetor de estado. As variáveis de estado associadas ao sistema do Pêndulo Inverso são: a posição x e a velocidade do carro v, o ângulo ϕ e a velocidade

angular w do pêndulo. Adicionalmente existe uma variável de controle F que representa a força aplicada no carro para manter o pêndulo na posição vertical.

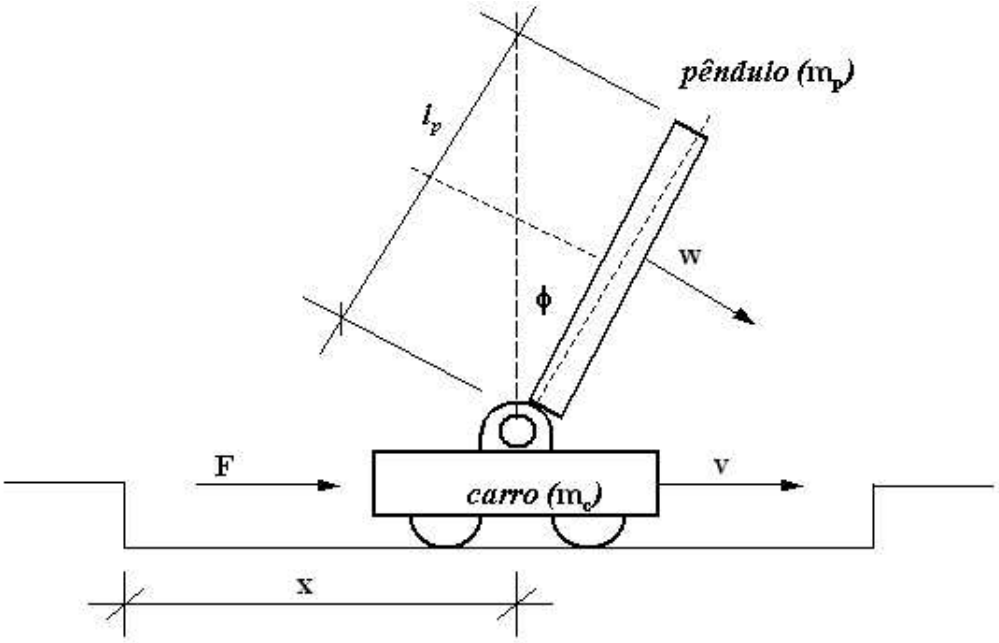

FIGURA 5.1 – Visualização gráfica do Pêndulo Inverso.

A Figura 5.1 apresenta a visualização de um Pêndulo Inverso com carro, sendo que seu comportamento dinâmico é descrito matematicamente pelas equações do processo (5.1), (5.2), (5.3) e (5.4) [10].

$$\dot{w} = \frac{k_1 sen(\phi) - k_2 w^2 sen(\phi) cos(\phi) - F cos(\phi)}{k_3 - k_2 cos(\phi) cos(\phi)} \tag{5.1}$$

$$\dot{\phi} = w \tag{5.2}$$

$$\dot{v} = k_3 F + k_2 k_3 w^2 sen(\phi) - k_2 k_1 cos(\phi) sen(\phi) k_4 k_3 - k_4 k_2 cos(\phi) cos(\phi) \tag{5.3}$$

$$\dot{x} = v \tag{5.4}$$

onde $k_1 = (m_c + m_p)g$, $k_2 = m_p l_p$, $k_3 = \frac{4}{3} l_p (m_c + m_p)$, $k_4 = m_c + m_p$, g= $9.81 m/s^2$ denota a aceleração gravitacional, m_c= 1,0 $[kg]$ e m_p= 0,1 $[kg]$ massa do carro e do pêndulo, respectivamente, e l_p= 0,5 $[m]$ metade do comprimento do pêndulo. A dinâmica do pêndulo inverso é numericamente computada com a constante de tempo Δt = 0,01.

Na construção do simulador do Pêndulo Inverso utilizou-se as equações de Euler 5.5, 5.6, 5.7 e 5.8, para gerar os valores para o próximo estado do sistema, possibilitando a observação do comportamento do sistema.

$$x(t+1) = x(t) + v(t)\Delta t \tag{5.5}$$

$$v(t+1) = v(t) + \dot{v}\Delta t \tag{5.6}$$

$$\phi(t+1) = \phi(t) + w(t)\Delta t \tag{5.7}$$

$$w(t+1) = w(t) + \dot{w}\Delta t \tag{5.8}$$

Utilizou-se, também, um controlador linear que pode controlar tanto o ângulo (ϕ) como a posição (x), descrito em [10], mostrado na Eq. (5.9).

$$F = K_w * w + K_v * v + K_x * x + K_\phi * \phi \tag{5.9}$$

onde $K_w = 0,8N/(rad/s)$, $K_v = 0,5N/(m/s)$, $K_x = 0,3N/m$ e $K_\phi = 45N/rad$.

Dois tipos de arquitetura de Rede Neural foram utilizados para os experimentos com o Sistema do Pêndulo Inverso. Devido a este fato, os experimentos estão divididos em duas subseções, onde cada uma apresenta os resultados obtidos de acordo com o tipo de rede utilizada. Para os primeiros experimentos ([73], [74] e [12]), as Redes Neurais treinadas forneciam como saída os valores de $\dot{w}$ e $\dot{v}$ calculadas através da próprias equações do modelo. Nos experimentos executados para a segunda arquitetura, as saídas eram calculadas considerando-se o estado atual do sistema e a constante de tempo utilizada, ou seja, a rede não foi treinada para aprender as funções $\dot{w}$ e $\dot{v}$, mas os valores das variáveis de estado.

5.2.1 Resultados Obtidos para a Aplicação do Pêndulo Inverso (I)

Treinamento da Rede Neural

Para o treinamento da RNA, foram gerados valores randômicos para as 4 variáveis de estado do sistema e para a força de controle, sendo os valores de saída foram calculados através das equações descritas pelo modelo do Pêndulo Inverso. Foram gerados 1.000 dados correspondentes as variáveis de entrada e saída da rede, totalizando, assim, 1.000 padrões de treinamento. Considerou-se para isso, os intervalos possíveis de valores para as variáveis de estado do sistema.

Várias topologias de Rede Neural foram criadas e testadas, a fim de encontrar a que mais se adaptasse ao sistema. Duas redes individuais foram treinadas para calcular os valores das equações de $\dot{w}$ e $\dot{v}$, tendo, cada uma delas, 3 neurônios de entrada (F, w e ϕ) e um neurônio de saída respectivo a cada uma das equações. Os erros obtidos para as saídas $\dot{w}$ e $\dot{v}$, mais significativas para a RNA, após 5.000 épocas de treinamento, são mostrados na Tabela 5.1.

No treinamento e validação das redes utilizou-se a técnica *10-fold cross-validation*, e a ferramenta utilizada foi o programa SNNS ("*Stuttgart Neural Network Simulator*") criado na Universidade de Stuttgart [75]. Este também foi usado para transformar a RNA selecionada em um programa na linguagem C, utilizado posteriormente nos métodos de extração, a fim de obter os valores de ativação da camada oculta da rede.

Neurônios na Camada Oculta	Erro Médio Quadrático para $\dot{v}$	Erro Médio Quadrático para $\dot{w}$
5	0,402	0,022
10	0,105	0,014
15	0,062	0,002
20	0,033	0,001

TABELA 5.1 – Erro médio quadrático das saídas $\dot{v}$ e $\dot{w}$, encontrado no treinamento das redes neurais com 4 topologias diferentes.

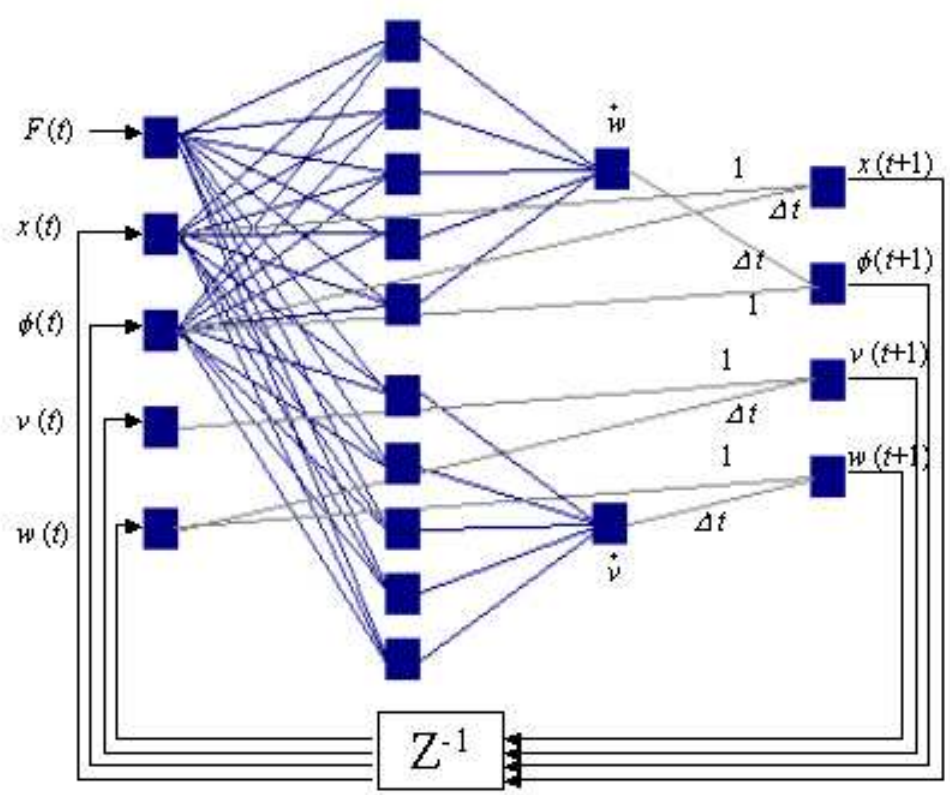

FIGURA 5.2 – Estrutura da RNR usada para aprender o comportamento do sistema dinâmico do Pêndulo Inverso. A rede possui 10 neurônios não-lineares na primeira camada oculta, sendo lineares os neurônios das demais camadas.

A arquitetura da RNA utilizada para aprender o comportamento do sistema dinâmico não linear do Pêndulo Inverso é apresentada na Figura 5.2. A rede possui 5 neurônios de entrada, que representam as 4 variáveis de estado (w, ϕ, v, x) e a variável de controle (F), 10 neurônios na camada escondida e 4 neurônios de saída, que representam as variáveis de estado no próximo passo de tempo. Como pode ser observado na figura, as redes treinadas para $\dot{w}$ e $\dot{v}$, com 5 neurônios na camada oculta, formam a arquitetura final da RNA utilizada, baseada na arquitetura da Rede

de Jordan. Embora os valores de erro, apresentados pelas redes com número maior de neurônios ocultos, fossem menores que os apresentadas para as redes com 5 neurônios, houve a necessidade de se utilizar o menor número de neurônios possível, tendo em vista o alto custo computacional apresentado para utilização dos métodos de extração. Outra razão para tal escolha, é a dificuldade encontrada na análise e interpretação de autômatos gerados com um grande número de estados.

Extração do Autômato Finito Difuso

Para a extração dos estados do autômato finito difuso, através da clusterização difusa, seis simulações foram executadas variando-se o ângulo inicial utilizado e a força aplicada para controlar o pêndulo. Através das simulações, foram geradas séries temporais para as seis simulações, totalizando 2.570 padrões, armazenados para serem utilizados como entrada para a Rede Neural. Os padrões obtidos das seis simulações foram apresentados à RNA treinada e suas ativações foram armazenadas e analisadas. A partir do comportamento apresentado por elas, construiu-se as funções de pertinência ($F's$) adequadas a cada neurônio da camada oculta.

As funções de pertinência ($G's$) utilizadas para representar os estados do AFDif foram calculadas através do "E" lógico difuso dos conjuntos difusos dos neurônios individuais da Rede Neural. Dentro de cada G a Rede Neural pode ser substituída por um sistema de equações lineares, portanto todas as ferramentas de controle linear são aplicáveis a cada estado individualmente para descrever as características do estado.

As simulações geradas para a obtenção do autômato iniciam com 6 valores diferentes de ângulo do pêndulo, com a utilização do controlador as simulações iniciam com ângulos respectivamente de -42°, -41°, 41° e 42°, e para as simulações sem controlador utilizou-se -0,1° e 0,1°, ou seja, levemente inclinado para o lado esquerdo e para o lado direito. Estes valores de ângulo foram escolhidos após a análise do sistema, efetuada através do simulador construído para o sistema do Pêndulo Inverso.

Através do cálculo das funções de pertinência, executado para as ativações das seis simulações, escolhidas por sua representatividade, foram obtidos gráficos, que possibilitam a visualização das transições entre cada estado. A Figura A.1 apresenta as funções de pertinência, referentes aos estados, para as seis simulações. De modo a melhorar a visualização, somente foram utilizados os 150 primeiros padrões de cada simulação, tendo em vista a melhor representatividade para a observação das transições. Por exemplo, no primeiro gráfico apresentado na figura, observa-se a transição do estado 4 para o estado 5. Isso pode ser observado através do aumento da curva na cor azul, ou seja, da função de pertinência que representa o estado 5, e a diminuição na curva de cor rosa, representativa do estado 4. No Apêndice A, também podem ser observadas as seis transições de estado, plotadas em um mesmo gráfico, sendo as quatro primeiras extraídas a partir de simulações utilizando o controlador e as duas últimas sem a utilização de nenhuma força para manter o pêndulo na posição desejada.

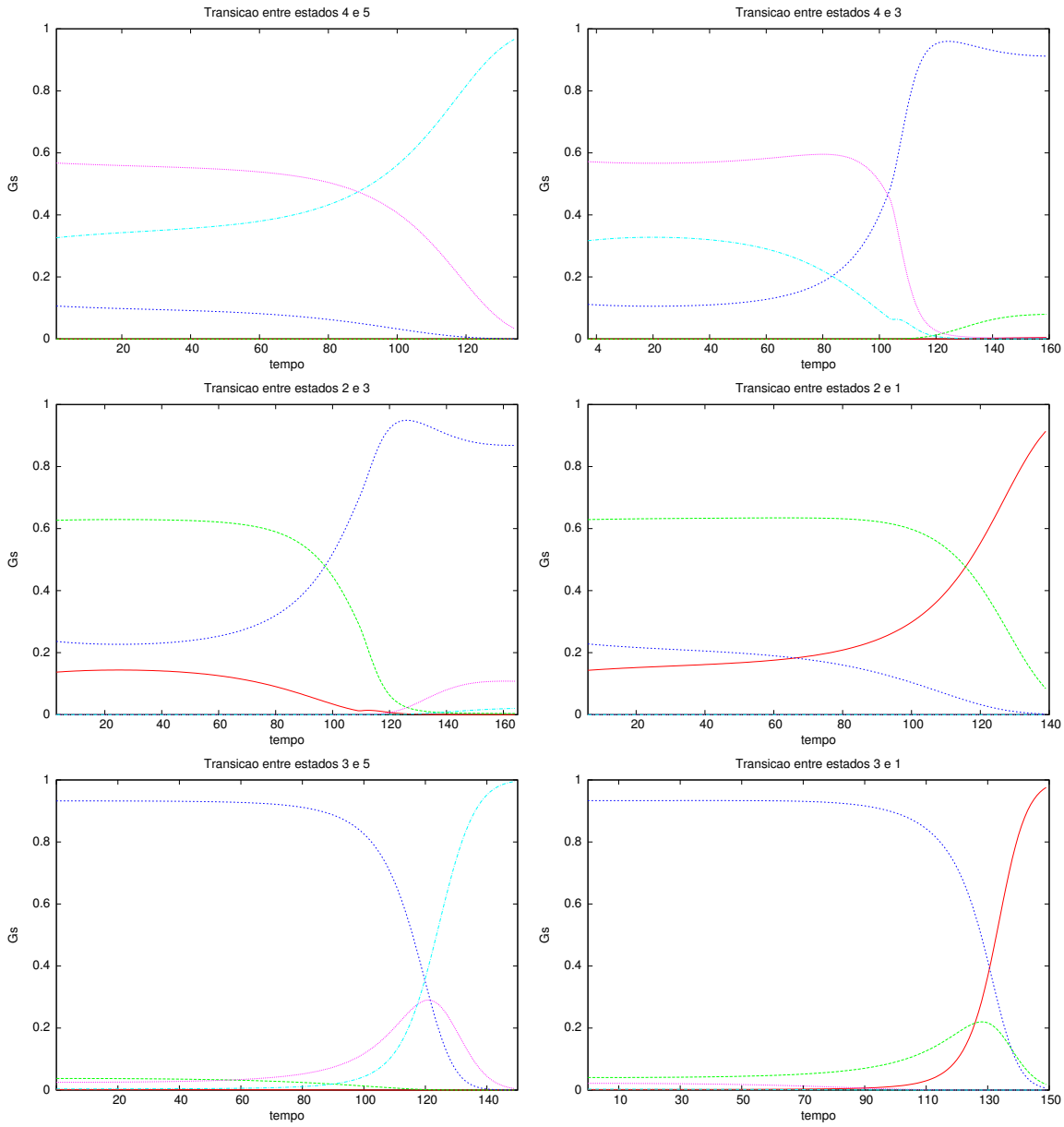

FIGURA 5.3 – Identificação das transições do AFDif para o sistema do Pêndulo Inverso, através das funções de pertinência dos estados. O primeiro gráfico (acima esquerda) apresenta a simulação iniciada com ângulo de -42°, usando o controlador, e identifica a transição entre os estados 4 e 5. No segundo (acima direita), com a simulação iniciada em -41°, usando o controlador, identifica-se a transição entre os estados 4 e 3. Terceiro (meio esquerda) e quarto (meio direita) gráficos, apresentam simulações iniciando 41° e 42°, usando o controlador, identificam-se as transições entre os estados 2 e 3, e 2 e 1. Quinto (abaixo esquerda) e sexto (abaixo direito) gráficos, apresentam simulações iniciadas -0,1° e 0,1°, sem o controlador, onde identificam-se as transições entre os estados 3 e 5, e 3 e 1.

De acordo com as duas primeiras simulações (-42° e -41°) (gráficos acima na Figura A.1), observa-se que o pêndulo inicia desequilibrado para a esquerda e poderá, ou transitar para o estado de equilíbrio, ou cair para o lado esquerdo. Já para as duas seguintes simulações (41° e 42°) (gráficos no meio na Figura A.1), o pêndulo inicia desequilibrado para a direita e poderá transitar para o estado de equilíbrio, ou cair para o lado direito. Nas duas últimas simulações (-0,1° e 0,1°) (gráficos abaixo na Figura A.1), onde nenhuma força de controle é aplicada, se a simulação for iniciada com o pêndulo pendendo ligeiramente para a esquerda ou para a direita, o sistema transitará diretamente do estado de equilíbrio para os estados de queda do pêndulo para o lado esquerdo ou para o lado direito. É importante observar nas duas últimas simulações, que na fase de transição de um estado ao outro, o sistema passa por um estado de equilíbrio, mas não chega a entrar totalmente neste estado (veja gráfico abaixo na Figura A.1, tempo aproximadamente 120 e 130).

O AFDif extraído a partir da Rede Neural utilizada para aprender o comportamento do sistema do pêndulo inverso é apresentado na Figura 5.4 na forma de um diagrama de transições de estados. Cinco estados foram identificados para o sistema, e as transições entre eles são determinadas através da força aplicada ao sistema e do ângulo inicial utilizado nas simulações executadas para construção do autômato finito. Os estados apresentados no autômato finito da Figura 5.4 são descritos conforme a Tabela 5.2.

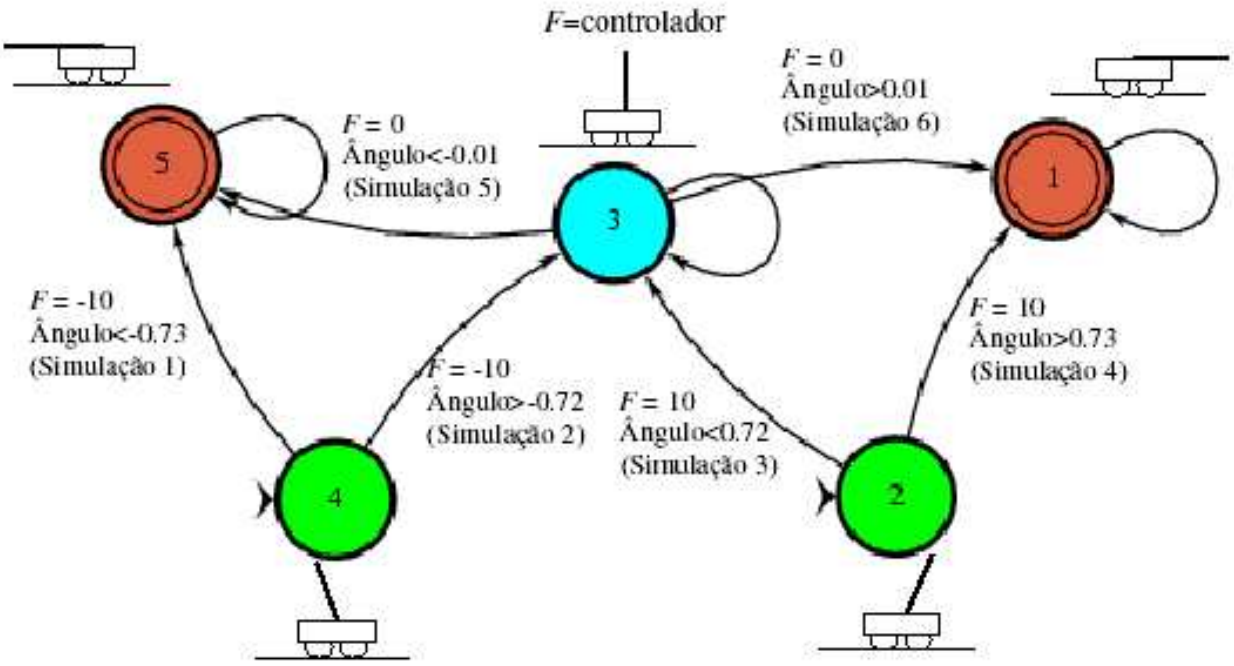

FIGURA 5.4 – Autômato Finito para o Sistema do Pêndulo Inverso.

Estados	Descrição
1	Pêndulo inclinado 90° para o lado direito
2	Pêndulo levemente inclinado para o lado direito
3	Pêndulo equilibrado
4	Pêndulo levemente inclinado para o lado esquerdo
5	Pêndulo inclinado 90° para o lado esquerdo

TABELA 5.2 – Descrição dos estados do Autômato Finito do Pêndulo Inverso.

Utilizando o simulador construído, fez-se uma comparação de performance entre os três modelos que apresentam o conhecimento do sistema do Pêndulo Inverso, as equações dos sistema, a Rede Neural treinada e o AFDif extraído. Para as três simulações, uma com cada modelo, foram consideradas as mesmas condições iniciais, ou seja, as simulações foram iniciadas com os valores das variáveis de estado: $x = 0$, $\phi = -40°$, $v = 0$, $w = 0$, $F = 0$ e utilizou-se o mesmo tempo de simulação. A configuração inicial e final das simulações pode ser observada na Figura 5.5, capturada a partir do simulador gráfico *GLManipulator* [35], usado para executar a simulação gráfica do sistema.

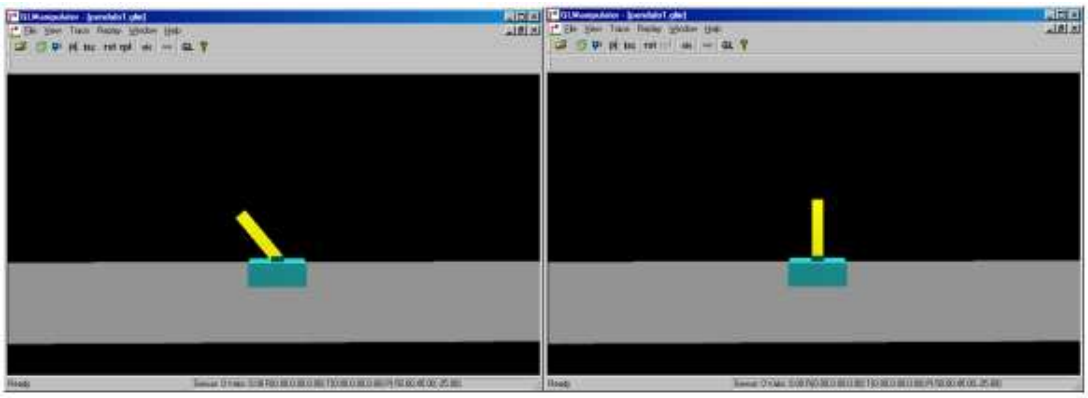

FIGURA 5.5 – Visualização gráfica do Pêndulo Inverso. A figura à esquerda apresenta a posição inicial do pêndulo usada na simulação, e a figura à direita apresenta a posição final desejada ao final da simulação.

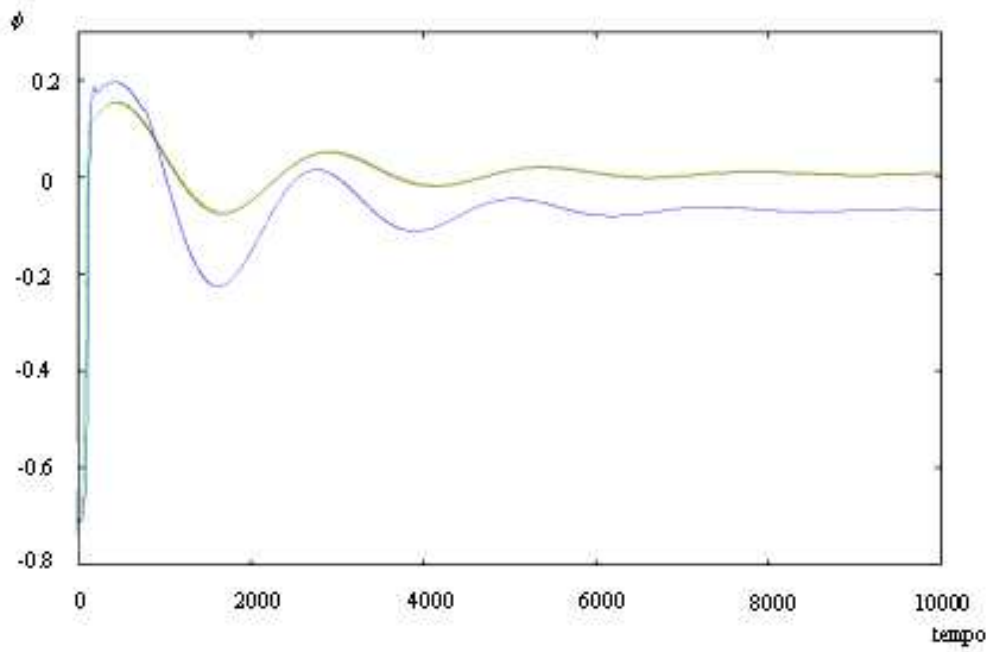

FIGURA 5.6 – Gráfico do ângulo do Pêndulo Inverso durante a simulação com as equações do sistema (cor vermelha), com a Rede Neural treinada (cor verde) e com o AFDif (cor azul).

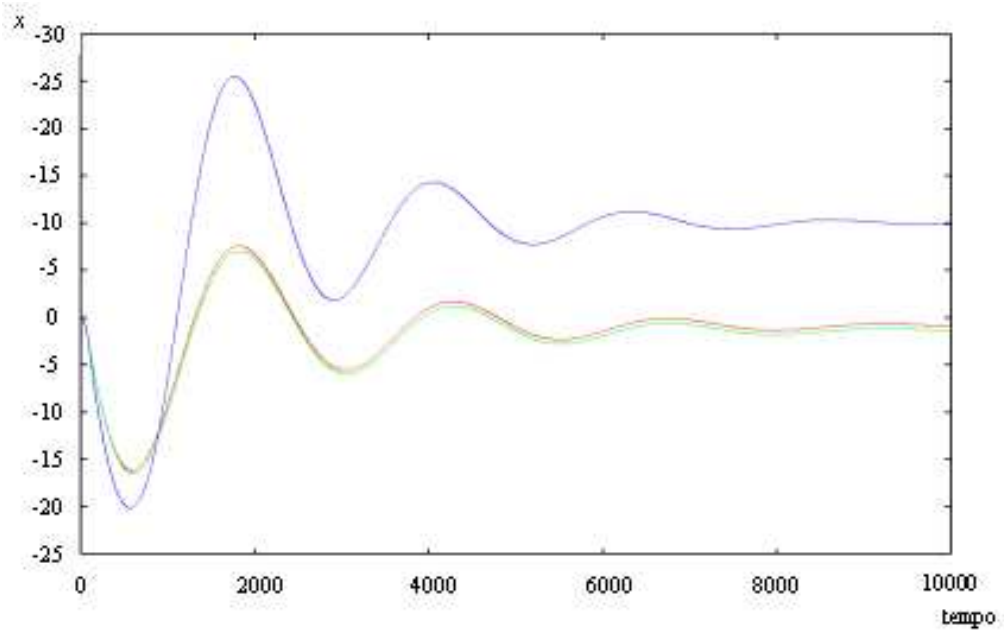

FIGURA 5.7 – Gráficos da posição do Pêndulo Inverso durante a simulação com as equações do sistema (cor vermelha), com a Rede Neural treinada (cor verde) e com o AFDif (cor azul).

Nos gráficos apresentados nas Figura 5.6 e 5.7 pode-se observar as três simulações através dos valores das variáveis x, ϕ, validando, desta forma, tanto o comportamento aprendido pela Rede Neural, como o comportamento demostrado no autômato finito difuso. A diferença encontrada na simulação com o AFDif deve-se ao fato, de que este formalismo representa uma generalização do conhecimento contido na Rede Neural, ou seja, o AFDif extraído é um modelo bem mais geral, que apresenta menor detalhamento que a rede e principalmente que as equações do modelo.

Extração do Autômato Finito Determinístico

Utilizando o método de clusterização *K-means*, foram clusterizados os dados utilizados para o treinamento da Rede Neural, em 5, 10 e 30 clusters, para analisar o efeito do número de estados na precisão do autômato. Após a clusterização, os mesmos dados de treinamento serviram como entradas para o simulador, gerando os valores das variáveis de estado desejadas no próximo instante de tempo. Este valores foram classificados de acordo com os clusters formados, a fim de identificar os estados aos quais pertencem e identificar possíveis transições entre os estados.

Além dos 1.000 padrões de treinamento, mais 100.000 padrões de entrada foram gerados e simulados, para que todo o espaço fosse devidamente amostrado, e que possíveis transições entre estados pudessem ser identificadas. Utilizando 5 ou 10 centróides para a clusterização com *K-means* não foram obtidos resultados satisfatórios. As transições obtidas para 5 e 10 estados não conseguiam representar o comportamento encontrado no pêndulo ou apresentavam comportamentos que o sistema original não poderia demonstrar, como por exemplo, ocorrer uma transição entre dois estados que possuíam os valores de ângulo para os centróides onde o pêndulo estava caído para ambos os lados.

Optou-se então por aumentar o número de estados escolhidos para a clusterização, com o intuito de que, se os estados do autômato representassem uma região menor do espaço de estados, seria possível visualizar transições que demonstrassem o comportamento do sistema. Para a clusterização usando 30 centróides (estados do autômato), foi gerado um arquivo de configuração para ser apresentado à ferramenta *FSA Utilities Toolbox* [59], que constrói automaticamente a visualização do autômato finito na forma de um diagrama de transições de estados. Para a construção do autômato levou-se em consideração, evidentemente, as transições encontradas em os 30 estados gerados. A visualização deste AFDet pode ser observada na Figura 5.8.

Considerando a complexidade apresentada pelo autômato com 30 estados, foram selecionados e interpretados 4 destes estados, sendo seus centróides especificados na Tabela 5.3. De acordo com esta tabela contendo os valores dos centróides e a figura contendo o autômato, se o pêndulo tiver um ângulo próximo de 0,14rad (estado 10) e a força próxima de -7N é aplicada (para a esquerda), irá existir uma transição para o estado 3 ($\tilde{}$0,35rad). Na direção oposta, irá existir uma transição do estado 11 (levemente inclinado para a direita) para o estado 13 (levemente inclinado para a esquerda) após uma força de aproximadamente $+7N$ ser aplicada. A mudança na velocidade angular pode ser explicada de maneira similar.

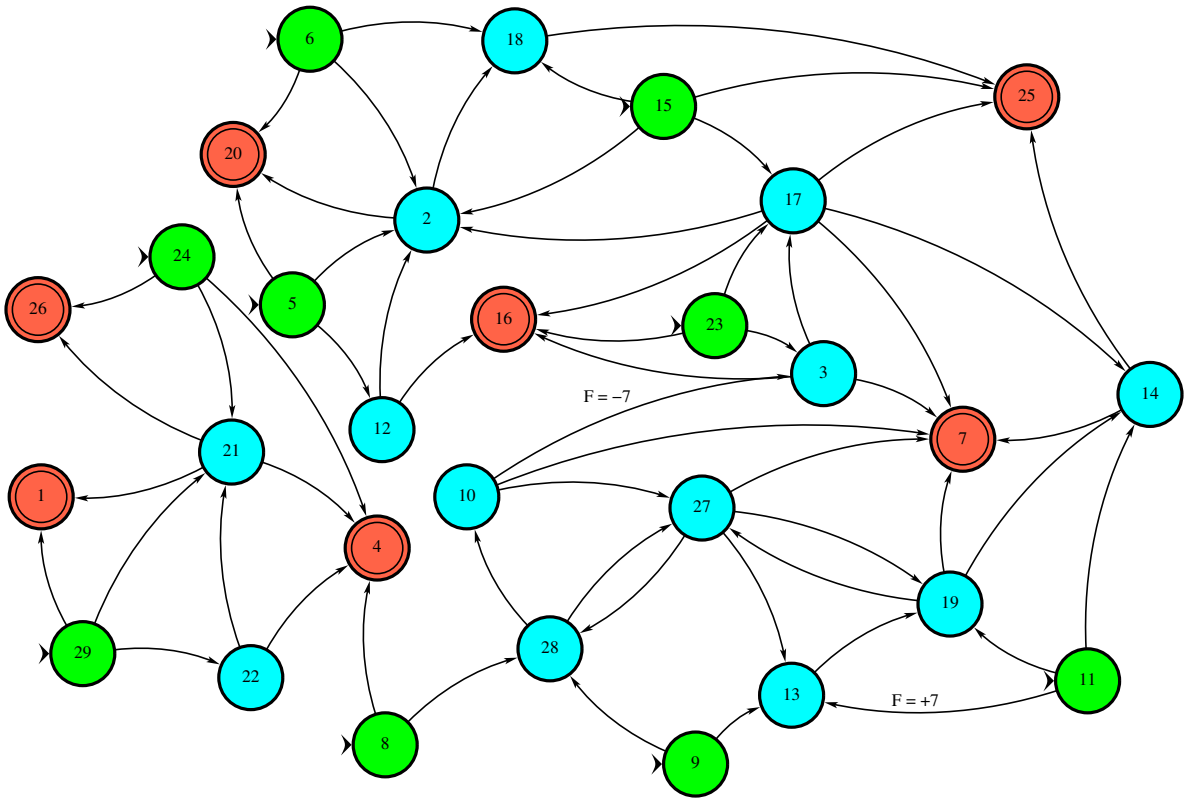

FIGURA 5.8 – Autômato Finito gerado pela ferramenta FSA, utilizando o método de clusterização do *K-means*, com representação de 30 estados para o modelo do Pêndulo Inverso.

Estado	**Ângulo** (ϕ[rad])	**Força** ($F[N]$)	**Velocidade Angular** ($w[rad/s]$)
3	0.3589 rad = 20.6°	-7,2209	0,3330
10	-0.1487	-6,2144	0,0306
11	0.1387	7,2926	1,0976
13	-0.2788	7,8580	0,0865

TABELA 5.3 – Centróides para 4 estados do autômato mostrado na Figura 5.8.

O algoritmo *K-means* para a clusterização das ativações apresentou a possibilidade de escolha do número de estados que se desejava representar no autômato finito. Embora adequada para a representação dos estados, esta técnica não se mostrou atrativa para a extração de conhecimento a partir da Rede Neural, pois quanto menor a quantidade de centróides escolhidos, ou seja de estados escolhidos, menor a representatividade fornecida pelos clusters, e quanto maior o número de centróides, maior era dificuldade de entendimento do modelo a partir da representação

em forma de autômato. Uma demonstração clara desta complexidade pode ser visualizada no autômato representado na Figura 5.8, onde são utilizados 30 estados para representar o espaço de estados do Pêndulo Inverso.

De forma similar que para o autômato de estados difusos, foram apresentados para o *K-means* os valores das 6 simulações, iniciadas com ângulos de -42°, -41°, 41° e 42° para as simulações com controlador, e sem controlador com ângulos de -0,1° e 0,1°, totalizando 2.570 padrões. Observou-se que dos 30 estados inicialmente apresentados, somente 19 eram utilizados para representar os dados gerados com as simulações, ou seja, para representar o espaço amostrado durante as 6 simulações efetuadas.

A partir dos experimentos realizados observou-se que o número de estados para o método proposto é radicalmente menor que para a clusterização realizada com o método *K-means*. Apesar do número de estados para o método *K-means* poder ser escolhido *a priori*, uma região do espaço de estados é representada por um vetor (centróide), o que implica uma redução drástica na quantidade de informação sobre o estado. Em contrapartida, para os estados difusos, a informação sobre um estado não é representada por um centróide, mas por uma relação linear entre as variáveis de estado, mais precisamente, entre $\phi(t+1)$, $w(t+1)$, $x(t+1)$, $v(t+1)$ e $\phi(t)$, $w(t)$, $x(t)$, $v(t)$ e $F(t)$. Isso torna a representação do estado mais complexa, ou seja, através de uma matriz que descreve a relação entre as variáveis de estado e não do vetor que representa o centróide, porém esta metodologia trás vantagens em termos de redução no número de estados e portanto em termos de compreensibilidade do autômato gerado.

5.2.2 Resultados do sistema do Pêndulo Inverso (II)

Treinamento da Rede Neural

Nesta segunda fase de experimentos, a Rede Neural é treinada utilizando-se séries temporais geradas a partir das equações do modelo. Isso a difere dos primerios experimentos realizados, visto que, para aqueles, foram usados valores gerados randomicamente. Nos novos experimentos, os dados para o treinamento da Rede Neural foram gerados utilizando as equações de Euler (5.5), (5.6), (5.7) e (5.8), para obter os valores do próximo passo de tempo para as variáveis de estado de acordo com seus estados iniciais. Foram geradas 101 séries temporais, totalizando 10.010 padrões.

Através de simulações do modelo do sistema do Pêndulo Inverso, pode-se estimar possíveis posições iniciais que representassem adequadamente o espaço de estados do sistema. Os valores iniciais utilizados para as variáveis de estado foram: $x = [-3, -2, -1, 0, 1, 2, 3]$; $\phi = [-80°, -60°, -40°, -30°, -20°, -10°, 0°, 10°, 20°, 30°, 40°, 60°, 80°]$; $v = 0,0$ e $w = 0,0$. Para obter a menor interferência possível da força de controle nas variáveis de estado, utilizou-se valores fixos para F e não mais o controlador apresentado na descrição do sistema. Os valores utilizados para a variável de controle foram $F = [-10, -8, -6, -4, -2, 0, 2, 4, 6, 8, 10]$. Com estes valores iniciais foi possível cobrir todo o espaço de estados do sistema. A dinâmica do Pêndulo Inverso foi computada com a constante de tempo $\Delta t = 0,01$.

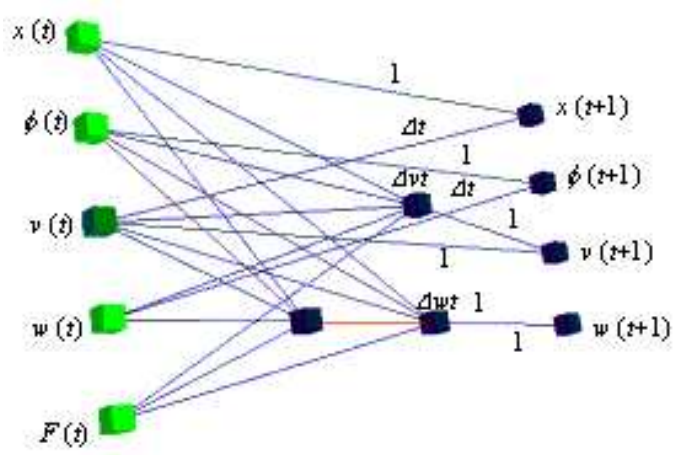

FIGURA 5.9 – Estrutura da Rede Neural usada para aprender o comportamento do sistema dinâmico do Pêndulo Inverso. A rede possui 5 neurônios de entrada, que representam as 4 variáveis de estado (x, ϕ, v, w) e a variável de controle (F), e 4 neurônios de saída que representam as variáveis de estado no próximo passo de tempo.

Baseou-se a construção da Rede Neural, na arquitetura da rede de Jordan, para que o comportamento não linear apresentado pelo sistema dinâmico pudesse ser aprendido. Várias topologias foram testadas para a Rede Neural a fim de encontrar a que apresentasse a melhor generalização dos dados, e que, quando usada para a simulação, representasse o comportamento do sistema. Para o treinamento, optou-se por utilizar duas redes independentes para aprender as não linearidades apresentas pelas variáveis de velocidade linear e velocidade angular, sendo estas, as variáveis que apresentavam não linearidades no sistema. As variáveis de estado e de controle foram aplicadas como entrada para a redes e suas saídas foram os valores de Δwt e Δvt (Ver Figura 5.9). Estes valores foram obtidos subtraindo o valor do estado atual, do próximo estado, ou seja: $\Delta wt = w(t+1) - w(t)$ e $\Delta vt = v(t+1) - v(t)$.

A técnica *10-fold cross-validation* foi usada para treinamento e validação das diferentes topologias de Redes Neurais testadas. Os erros mais significantes obtidos para as saídas das redes treinadas, após 2.000 épocas, são mostrados na Tabela 5.4.

Topologia da Rede	**5-0-1**	**5-1-1**	**5-2-1**	**5-4-1**	**5-8-1**
MSE para Δvt	0,000098	0,000089	0,000001	0,000014	0,000078
MSE para Δwt	0,078464	0,069346	0,001389	0,000964	0,000832

TABELA 5.4 – Erro médio quadrático das saídas Δvt e Δwt. Valores encontrados para o treinamento das Redes Neurais com 5 diferentes topologias.

Considerando os erros apresentados por cada uma das redes treinadas e o comportamento demonstrado pelo sistema através da simulaçao utilizando-se cada uma delas, selecionou-se a

arquitetura, mostrada na Figura 5.9, para ser utilizada na extração dos três formalismos de representação do conhecimento abordados pelo trabalho. Desta forma, as redes treinadas que foram usadas para formar a arquitetura final da RNR são, a rede 5-0-1 para Δvt e 5-1-1 para Δwt. A utilização de somente um neurônio não linear na camada oculta da rede, deve-se à análise dos erros e também à constatação de que esta topologia, quando usada para simulação, consegue representar o comportamento do sistema de forma satisfatória. Esta rede se tornou bastante atraente para as extrações, pois foi possível identificar facilmente o número de estados necessários para a representação do comportamento do sistema, além do fato de poder trabalhar com um número pequeno de estados, que representa diminuir a complexidade obtida quando da necessidade de se utilizar muitos estados para a representação na forma de um AFDif.

Através da análise da função de ativação do neurônio oculto, como pode ser observado na Figura 5.10, identificou-se que o sistema pode ser representado através de três estados de um autômato, já que a função está sendo ocupada em toda a sua extensão, e pode ser aproximada através de três equações lineares. Desta forma, os três tipos de extração de conhecimento, apresentados a seguir, levarão esta análise em consideração.

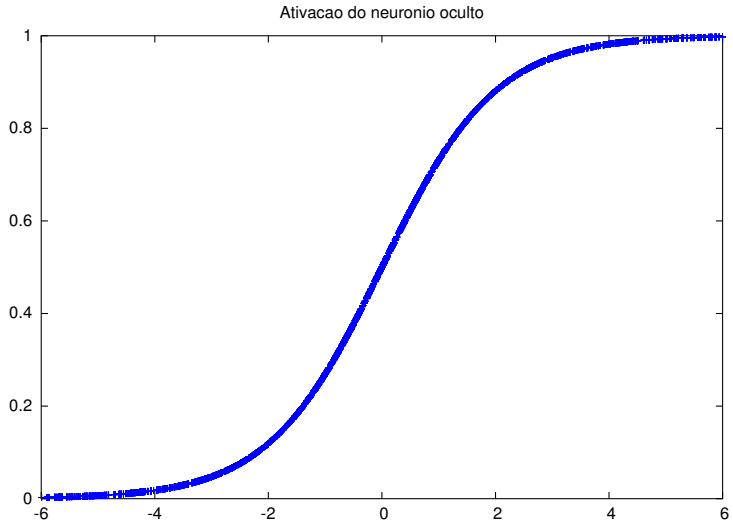

FIGURA 5.10 – Função de ativação do neurônio oculto da Rede Neural do Sistema do Pêndulo Inverso. Identificação da região de trabalho da função através das ativações dos dados gerados pela simulação do sistema.

Extração do Autômato Finito Difuso

A Figura 5.11 apresenta o autômato finito difuso extraído a partir da RNR com um neurônio não linear na camada oculta. De acordo com a figura, observa-se as possíveis transições existentes entre cada um dos três estados extraídos. Os desenhos associados às transições e aos estados, representam a configuração do sistema do pêndulo em cada ponto do autômato. O estado de

número 2, caracteriza a região do espaço de estados onde o pêndulo se mantém em equilíbrio, os estados 1 e 3, caracterizam, respectivamente, as regiões onde o pêndulo se encontra em desequilíbrio para o lado direito e para o lado esquerdo. Estando o pêndulo no estado 2, se a força aplicada possuir um valor negativo, o sistema irá passar para o estado 1, sendo necessário após esta transição, que uma força positiva seja aplicada ao sistema para conseguir manter o pêndulo em equilíbrio, de outra forma, o pêndulo tenderá a cair para o lado direito, levando o carro de onde está localizado para o lado esquerdo. A figura apresenta como referência para cada estado, os valores das variáveis de estado e força correspondentes aos seus centróides, sendo que, a partir da análise destes e das transições observadas para os dados gerados, torna-se possível extrair um autômato coerente com o sistema analisado.

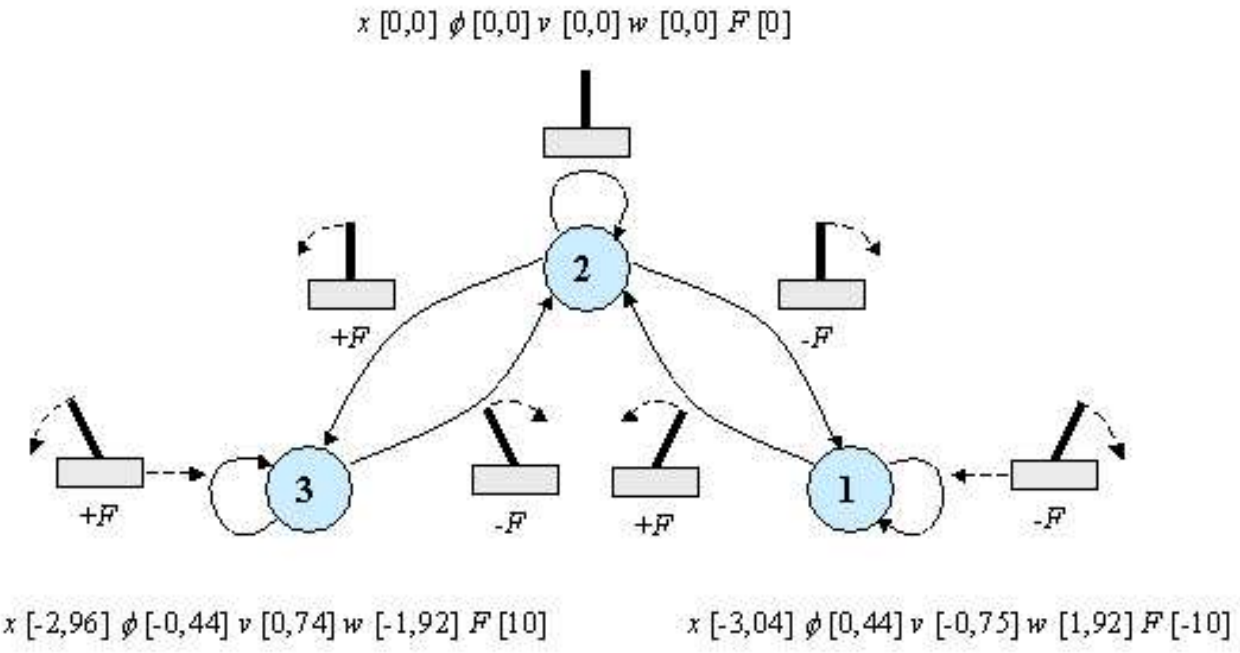

FIGURA 5.11 – Autômato Finito Difuso do Sistema do Pêndulo Inverso, representado através de um diagrama de transição de estados, extraído a partir da RNR com um neurônio na camada oculta.

As matrizes de transição que descrevem cada um dos estados do AFDif da Figura 5.11 são apresentadas na Tabela 5.5, considerando a Eq. 5.10.

$$\begin{bmatrix} x(t+1) \\ \phi(t+1) \\ v(t+1) \\ w(t+1) \end{bmatrix} = A * \begin{bmatrix} x(t) \\ \phi(t) \\ v(t) \\ w(t) \end{bmatrix} + B * [F(t)] + C \tag{5.10}$$

A integração das matrizes de transição requerem o cálculo da matriz Jacobiana, que representa a influência das componentes do vetor de estado na dinâmica do sistema. Para o cálculo

das matrizes apresentadas na Tabela 5.5, foram utilizados os valores de centróide de cada um dos estados. Na Eq. 5.10, A representa a matriz Jacobiana no centróide para as variáveis de estado, e B representa a matriz Jacobiana no centróide para a variável de controle. Para a obtenção da matriz C, deve-se substituir na Eq. 5.10 os valores dos centróides no tempo (t), seus valores no próximo passo de tempo $(t+1)$, e os valores das matrizes A e B já calculadas.

Estado	**Matriz A**				**Matriz B**	**Matriz C**
1	1	0	0,009990	0	0	−0,007513
	0	1	0	0,009999	0	−0,0785411
	0	−0,001192	0,999589	−0,000137	0,007331	−0,00897198
	0	0,050426	0,019884	1,00839	−0,007534	0,0862841
2	1	0	0,01	0	0	0
	0	1	0	0,01	0	0
	0	−0,001194	0,99959	−0,000136	0,007331	−0,000331
	0	0,185113	0,001297	1,003355	−0,014746	0,000022
3	1	0	0,009990	0	0	0,000007338
	0	1	0	0,009999	0	−0,000002047
	0	−0,001192	0,999595	−0,000131	0,007331	−0,000323946
	0	0,052500	0,019860	1,008392	−0,007558	−0,0822521

TABELA 5.5 – Matrizes de transição respectivas aos estados do AFDif extraído para o sistema do Pêndulo Inverso.

Comparando as simulações executadas utilizando as equações do sistema, a RNR treinada, e o autômato finito difuso, pode-se observar a semelhança de comportamento mostrada pelos três modelos de representação para o sistema do pêndulo inverso (Ver Figuras 5.12 e 5.13). Nota-se também, que a maior diferença apresentada entre as simulações, tanto no ângulo como na posição do carro que sustenta o pêndulo, está na utilização do AFDif extraído, tendo em vista uma maior generalidade apresentada pelo modelo.

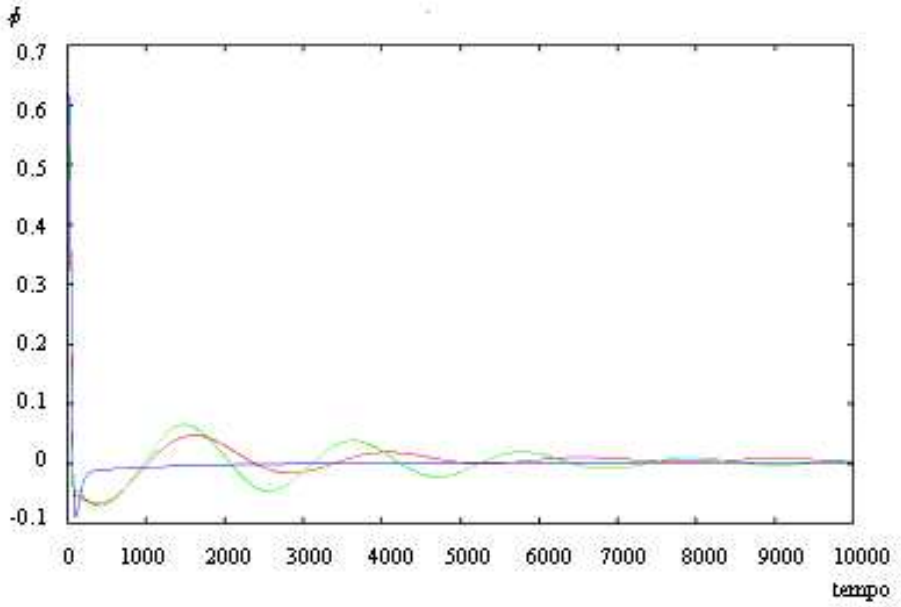

FIGURA 5.12 – Gráfico do ângulo do Pêndulo Inverso durante a simulação com as equações do sistema (cor vermelha), a Rede Neural treinada (cor verde) e o AFDif (cor azul).

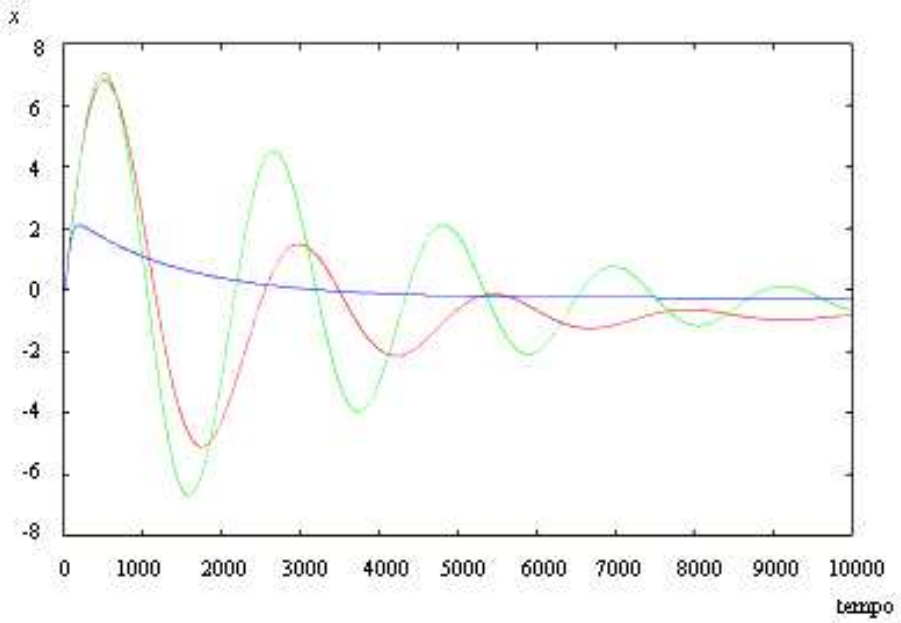

FIGURA 5.13 – Gráficos da posição do Pêndulo Inverso durante a simulação com as equações do sistema (cor vermelha), a Rede Neural treinada (cor verde) e o AFDif (cor azul).

Extração da Cadeia de Markov

Considerando os resultados obtidos com o treinamento das redes e a análise da função de ativação do neurônio da camada oculta, para a extração das cadeias de Markov e do autômato finito determinístico, optou-se por utilizar três clusters na clusterização com *K-means*. Como a rede apresenta somente um neurônio oculto, realizou-se a clusterização dos valores da única ativação, coletados a partir da apresentação dos dados de treinamento à rede selecionada. O resultado da cluterização pode ser visto no gráfico da Figura 5.14, onde cada cor determina os dados identificados para cada cluster. Ambos os eixos x e y do gráfico referem-se aos valores de ativação usados na clusterização.

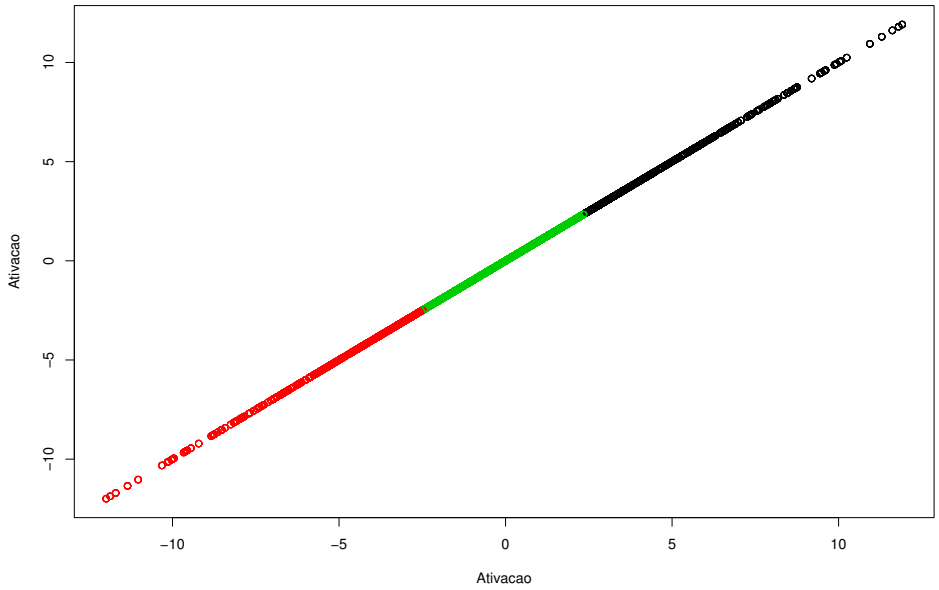

FIGURA 5.14 – Clusterização com K-means utilizando os valores de ativação do neurônio da camada oculta. Os eixos x e y do gráfico correspondem aos valores de ativação no neurônio.

Para a extração de cadeias de Markov optou-se por construir um cadeia para cada valor de força, para que fosse possível identificar as diferentes transições obtidas de acordo com a força aplicada, já que esta não faz parte das variáveis de estado do sistema. Inicialmente houve a clusterização dos valores de ativação do neurônio da camada oculta, para todos os dados de treinamento. Além das transições serem identificadas, foi necessário selecionar quais destas ocorriam com cada um dos valores de força apresentados pelos dados, ou seja, obter as freqüências com que cada transição ocorria. Através destas freqüências de transições calculou-se os valores de probabilidade de cada uma delas ocorrer e atribuíu-se estes valores as cadeias de Markov, representadas através de grafos na Figura 5.15. Pode ser observado na figura os valores do ângulo para os centróides de cada estado. Os estados coloridos com a cor verde apresentam somente transições para eles mesmos ou para outros estados, ou seja, são estados iniciais. Já os estados coloridos com a cor azul, apresentam transições vindas de outros estados, transições no próprio estado e para outros estados.

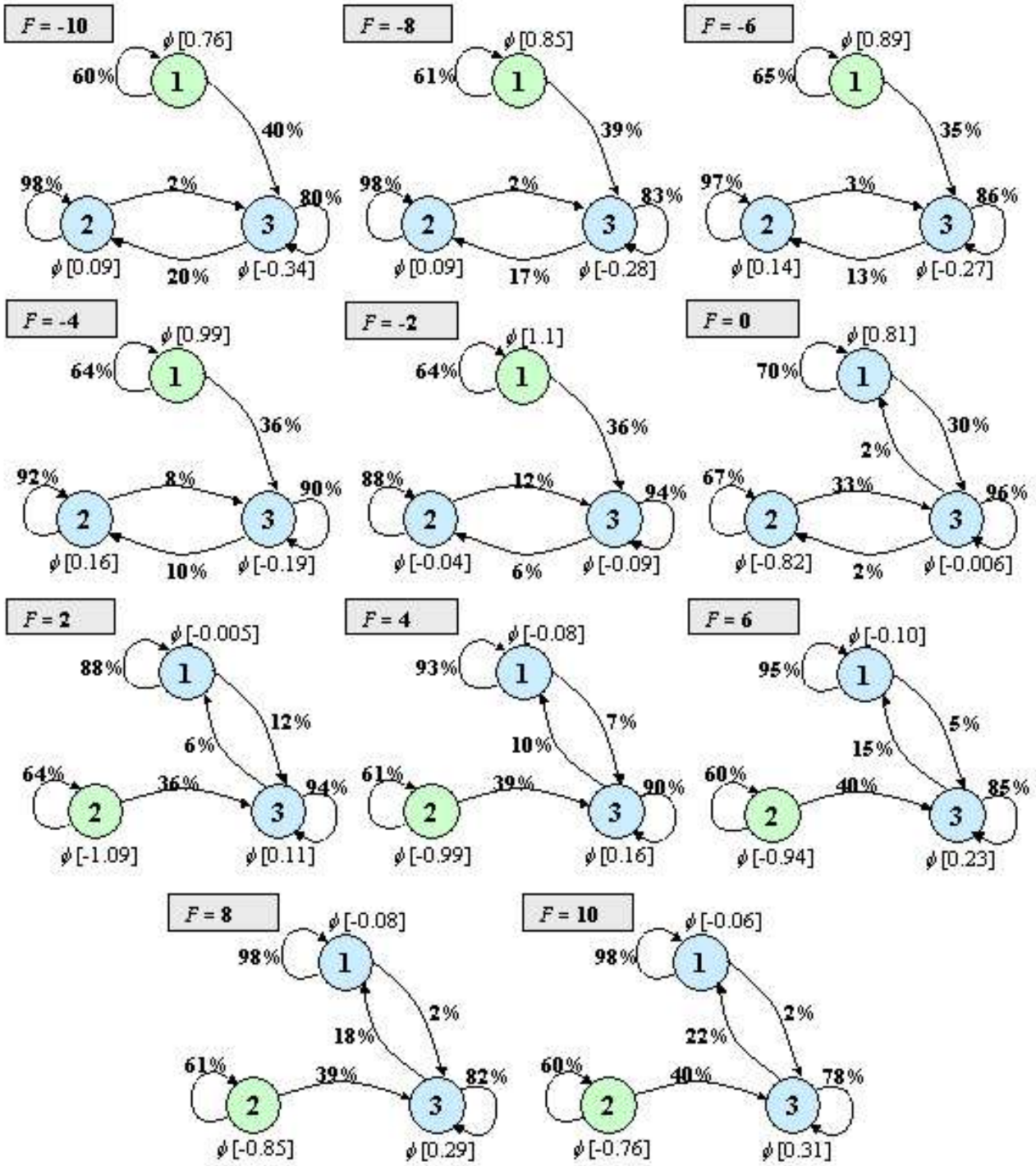

FIGURA 5.15 – Cadeias de Markov construídas para o sistema do Pêndulo Inverso.

Analisando as cadeias de Markov, pode-se caracterizar os estados extraídos através dos valores das variáveis de estado de cada centróide obtido. Os valores respectivos a cada centróide encontrado nas cadeias de Markov da Figura 5.15, estão descritos no Apêndice B. Na cadeia de Markov, onde não há força sendo aplicada ao sistema, o estado 1 representa a região onde o pêndulo está inclinado para o lado direito, o estado 2 representa a região onde o pêndulo está inclinado para o lado esquerdo, e o estado 3 pode ser considerado a região central, onde o pêndulo pode ser mantido em equilíbrio.

Como foram utilizados poucos estados para a clusterização, cada um deles representa um grande região do espaço de estados, o que pode ser observado pelos altos valores de probabilidade associados às transições que levam do estado para ele mesmo, possibilitando que se encontrem transições que não podem ser explicadas somente considerando a descrição dos centróides. Por exemplo, estando o sistema no estado 1, onde o pêndulo encontra-se inclinado para o lado direito, com uma força negativa sendo aplicada, a probabilidade mais elevada é que o sistema permaneça neste estado, embora exista a possibilidade de passar para o estado 3.

Extração do Autômato Finito Determinístico

A Figura 5.16 apresenta o AFDet extraído para o sistema do Pêndulo Inverso utilizando a mesma clusterização usada para a extração das cadeias de Markov. Através da análise dos centroídes e das transições encontradas quando da aplicação de cada uma das forças determinadas pelos dados de treinamento, obteve-se o AFDet mostrado na figura. Associado a cada estado está o valor do ângulo do centróide, e através dos desenhos representando o Pêndulo Inverso, determina-se os comportamentos encontrados em cada estado, de acordo com a força aplicada, considerando somente que a força seja positiva ou negativa. Conclui-se que a força apresenta forte influência sobre as variáveis de estado na clusterização, possibilitando que em todos os estados exista todos os valores de força possíveis. Analisando através do ângulo do pêndulo, pode-se observar que o estado 1 representa a região onde o pêndulo está da posição vertical até a posição inclinado para a direita, o estado 3 representa a região central, ou seja, a região onde o pêndulo mantém-se em equilíbrio, e o estado 2 representa desde a região central até a posição onde o pêndulo está inclinado para a esquerda. Assim, estando o sistema em qualquer um dos três estados, é possível que o pêndulo seja equilibrado, é claro, dependendo da força aplicada ao sistema.

Embora o AFDet extraído apresente os comportamentos característicos do Pêndulo Inverso, não se pode concluir precisamente qual o comportamento demonstrado por cada um dos estados e pelas suas transições, já que os mesmos valores de ângulo e força são representados por mais de um estado.

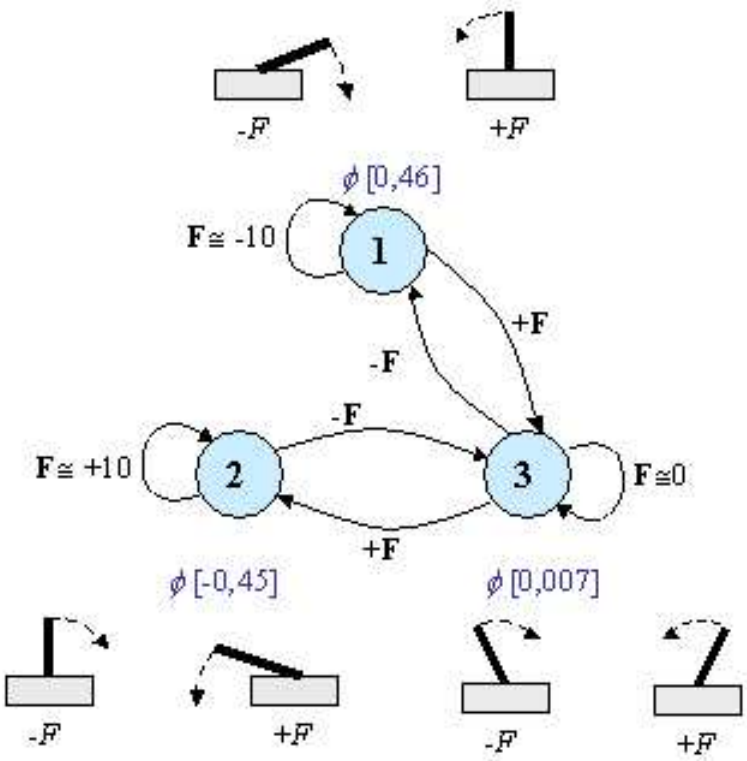

FIGURA 5.16 – Cadeia de Markov

Outros experimentos foram executados com o *K-means* utilizando também a Rede Neural com dois neurônios na camada oculta, sendo os resultados apresentados, semelhantes aos obtidos com somente um neurônio. Alguns destes resultados, como os gráficos das funções de ativação, o gráfico resultante da clusterização, as cadeias de Markov e os valores de seus respectivos centróides são apresentados no Apêndice C.

5.3 Sistema de Lorenz

As equações de Lorenz originaram-se a partir da construção de um simulador para a previsão do clima. Edward Lorenz (1963) programou um simulador que imprimia séries de números que representavam a evolução da pressão, temperatura, velocidade e direção do vento. Através da análise de um seqüência temporal, utilizando os números da série anterior como ponto de partida, Lorenz observou que as duas séries que deveriam ser aproximadamente iguais, após algumas simulações divergiram uma da outra e perderam qualquer semelhança. A causa deste efeito é que ao introduzir os números da simulação anterior, foram omitidas algumas casas decimais, sendo o suficiente para mudar completamente a evolução do sistema. Mais tarde chamou-se a este comportamento "Efeito Borboleta".

Este sistema é descrito pelas seguintes equações diferenciais não-lineares [62]:

$$\frac{dx}{dt} = -\sigma x + \sigma y \tag{5.11}$$

$$\frac{dy}{dt} = -xz + rx + y \tag{5.12}$$

$$\frac{dz}{dt} = xy - bz \tag{5.13}$$

onde σ, r e b são parâmetros que caracterizam as propriedades de um fluído e das condições térmicas e geométricas do sistema. A variável x representa a intensidade da convecção do fluido, y representa a diferença de temperatura entre as correntes do fluido e z representa a distorção do perfil de temperatura vertical. A FIgura 5.17 mostra órbitas típicas.

A solução numérica das equações com valores de parâmetros σ= 10, $r = 28$ e $b = 8/3$ conduz a um atrator caótico que pode ser visualizado em um espaço tri-dimensional com coordenadas x, y e z, conforme mostrado na Figura 5.17.

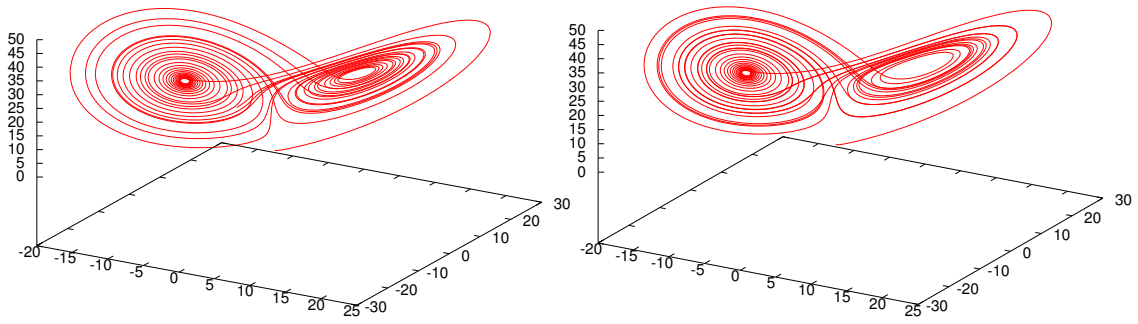

FIGURA 5.17 – Atrator de Lorenz após 10.000 passos de simulação com $\Delta t = 0,003$. O gráfico à esquerda mostra o atrator iniciando no ponto com coordenadas x=0, y=1 e z=0, e o gráfico à direita iniciando no ponto com coordenadas x=0,00001, y=1 e z=0.

O atrator é um conjunto com geometria fractal para o qual um sistema dinâmico converge, independentemente do ponto de partida [62]. Se cada estado das equações de Lorenz for representado por um ponto num gráfico tridimensional, pode-se verificar que estes convergem para um atrator tridimensional. Este é um atrator estranho, devido às suas propriedades geométrica e dinâmicas. O sistema é caótico, imprevisível, mas ao mesmo tempo converge para um atrator determinado.

O sistema de Lorenz é um sistema dinâmico não linear determinístico e, através de uma pequena mudança nas suas condições iniciais, é possível visualizar que o sistema exibe sensibilidade às condições iniciais. Representando em um gráfico, os valores das equações para x, y e z através do tempo, é possível observar como o sistema muda, com o passar do tempo, de acordo com o seu ponto inicial (Veja a Figura 5.18).

Através dos gráficos mostrados na Figura 5.18, pode-se observar que os valores para x, y e z, para os dois pontos iniciais, iniciam bastante aproximados, mas divergem rapidamente. Isso demostra como mudanças que parecem insignificantemente pequenas nas condições iniciais, podem afetar drasticamente o sistema.

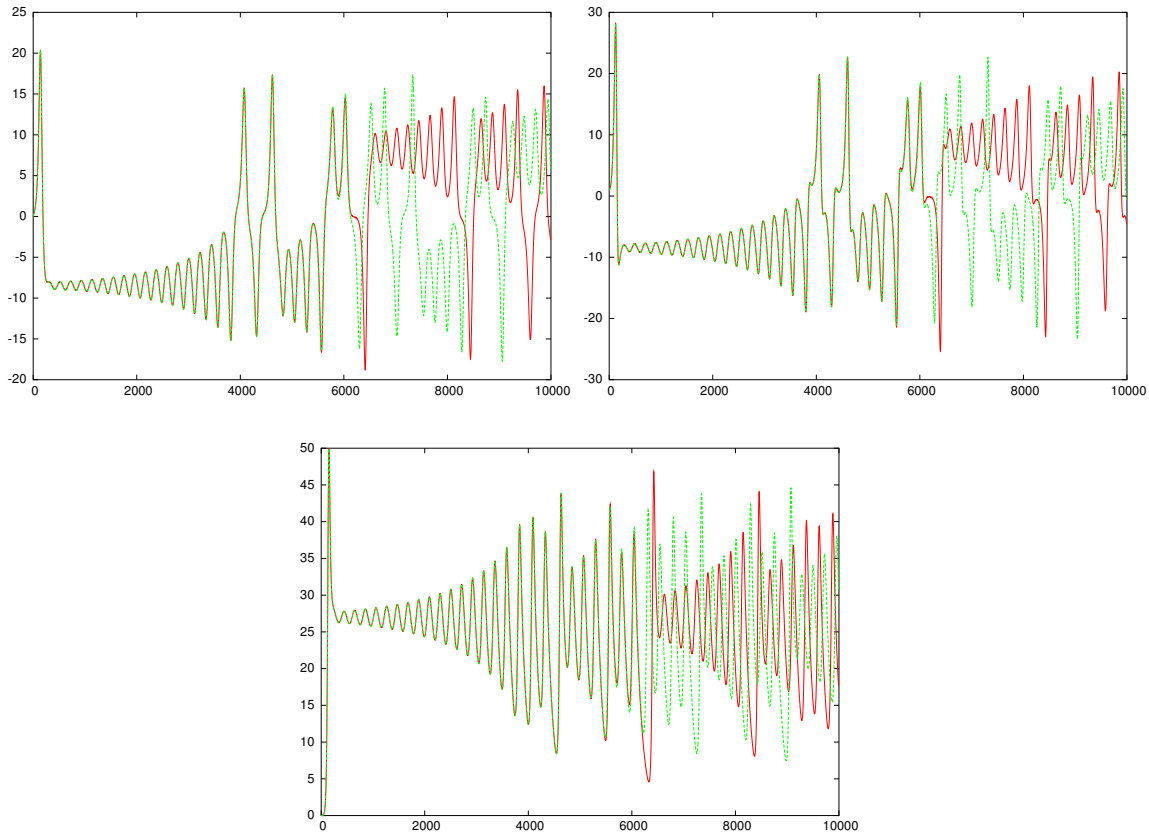

FIGURA 5.18 – Valores de x, y e z plotados através do tempo, gerados utilizando as condições iniciais (0, 1, 0) e (0,00001, 1, 0) no simulador para o sistema de Lorenz, durante um intervalo de 10.000 unidades de tempo, com passos de tempo de 0,003.

Treinamento da Rede Neural

Os dados utilizados para os experimentos com o Sistema de Lorenz foram gerados através da simulação das equações do sistema utilizando Runge-Kutta [11]. Inicialmente foram geradas seqüências temporais utilizando a constante de tempo $\Delta t = 0,003$, de forma a cobrir o espaço de estados não somente dentro do atrator. Foi gerada uma seqüência de 20.000 pontos iniciada por um ponto dentro do atrator, e 1.000 seqüências de 20 pontos iniciadas por pontos transientes escolhidos randomicamente.

Para a escolha do valor da constante de tempo adequado ao treinamento da Rede Neural, foi executada a análise da curva de autocorrelação do sistema, que pode ser usada como uma função

de aproximação de primeira ordem para o valor de passo de tempo [1]. A função de autocorrelação provê uma medida de similaridade de um sinal com sua versão atrasada no tempo. Através da análise das curvas de autocorrelação das seqüências de dados geradas, foi possível estimar um passo de tempo (Δt) de 0,06 para ser usado no treinamento das redes. Outro valor utilizado como passo de tempo para o treinamento foi de 0,03, que representa a metade do valor original.

Foram treinadas várias topologias de Rede Neural Recorrente, baseadas na rede de Jordan, apresentando 3 neurônios de entrada, 3 neurônios de saída e variando o número de neurônios em uma única camada oculta, para aprender o comportamento do Sistema de Lorenz. A Figura 5.19 mostra a estrutura de RNR utilizada, considerando 11 neurônios na camada oculta.

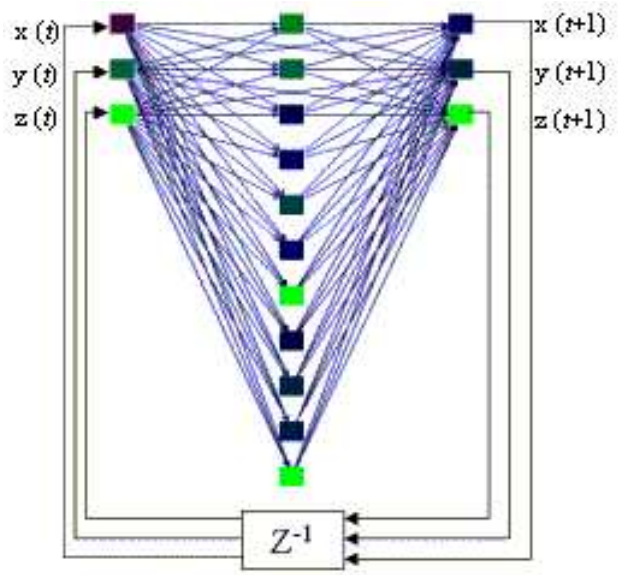

FIGURA 5.19 – Estrutura da RNR usada para aprender o comportamento do Sistema de Lorenz. Possui as 3 variáveis de estado (x, y e z) como entrada e seus valores no próximo passo de tempo são obtidos como saída.

A RNR, apresentada na figura, recebe como entrada o estado atual do sistema $[x(t), y(t), z(t)]$ e computa o valor do próximo estado do sistema $[x(t+1), y(t+1), z(t+1)]$ após um passo de tempo.

Para o treinamento das redes foram usados 2.000 padrões com $\Delta t = 0,06$ e 4.000 padrões com $\Delta t = 0,03$. A validação das RNRs foi efetuada com bases de teste contendo a metade da quantidade dos padrões existentes nas bases de treinamento, evidentemente, os dados usados para o teste são diferentes dos usados para o treinamento.

Após o treinamento, as redes foram utilizadas na simulação do sistema para que a topologia do atrator pudesse ser investigada. Todas as topologias, utilizadas para o treinamento, foram testadas com o simulador construído para o sistema, sendo que os resultados mais relevantes obtidos com a simulação podem ser vistos na Tabela 5.6, juntamente com os erros de treinamento das redes. Como pode ser observado na tabela, algumas redes não conseguiram aprender o

comportamento do sistema, ou seja, não conseguiram através de simulação, representar o atrator de Lorenz.

Topologia da RNR	$\Delta t = 0,03$		$\Delta t = 0,06$	
	MSE	Simulação	MSE	Simulação
3-3-3	12,07	NÃO	92,33	NÃO
3-10-3	0,99	NÃO	38,68	NÃO
3-11-3	0,49	SIM	17,71	NÃO
3-12-3	0,34	SIM	14,12	NÃO
3-13-3	0,43	SIM	12,40	NÃO
3-14-3	0,36	SIM	6,24	NÃO
3-15-3	0,24	SIM	5,78	NÃO
3-16-3	0,25	SIM	5,88	SIM
3-17-3	0,35	SIM	5,20	SIM
3-18-3	0,29	SIM	6,02	SIM
3-19-3	0,18	SIM	4,15	NÃO
3-20-3	0,16	SIM	2,94	SIM

TABELA 5.6 – Para cada Δt, é apresentado o erro médio quadrático do treinamento das redes e o resultado da simulação utilizando cada uma das topologias. O resultado da simulação é SIM, se a simulação com a RNR conseguiu reproduzir o atrator de Lorenz.

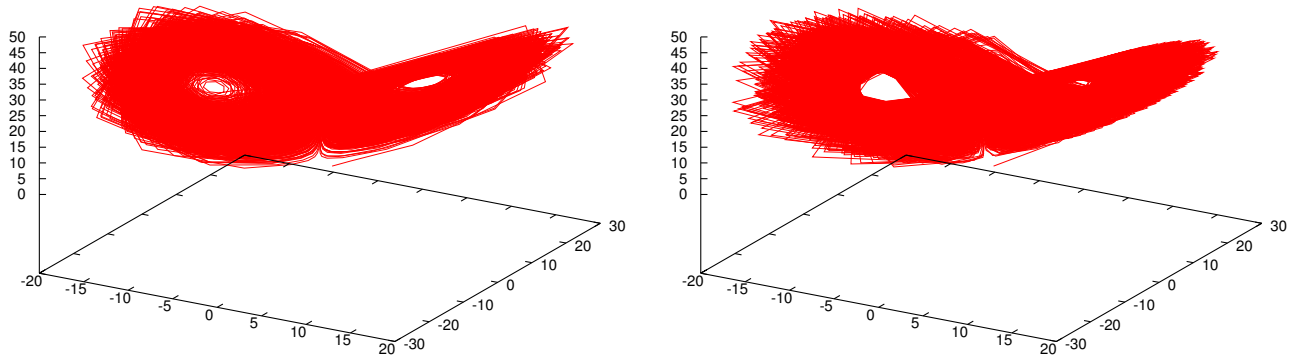

FIGURA 5.20 – Representação do atrator de Lorenz utilizando as RNRs. O gráfico à esquerda apresenta o atrator utilizando a rede com 11 neurônios na camada oculta, com $\Delta t = 0,03$, e o gráfico à direita apresenta o atrator utilizando a rede com 16 neurônios ocultos, com $\Delta t = 0,06$.

Como pode ser visto na Figura 5.20, o comportamento apresentado pelas simulações utilizando as Redes Neurais é similar ao comportamento apresentado pelas equações do sistema.

Nota-se uma diferença de discretização entre os modelos, contudo, isso não interfere no conhecimento representado.

Outra forma adotada para validação e escolha da Rede Neural utilizada para a extração de conhecimento, foi através da comparação entre a densidade de pontos localizados no espaço de estados do atrator, construído com as equações e com as Redes Neurais. Para isso, o gráfico 3D apresentado na Figura 5.21 foi dividido em 35 áreas, levando-se em consideração somente os eixos x e y para esta divisão.

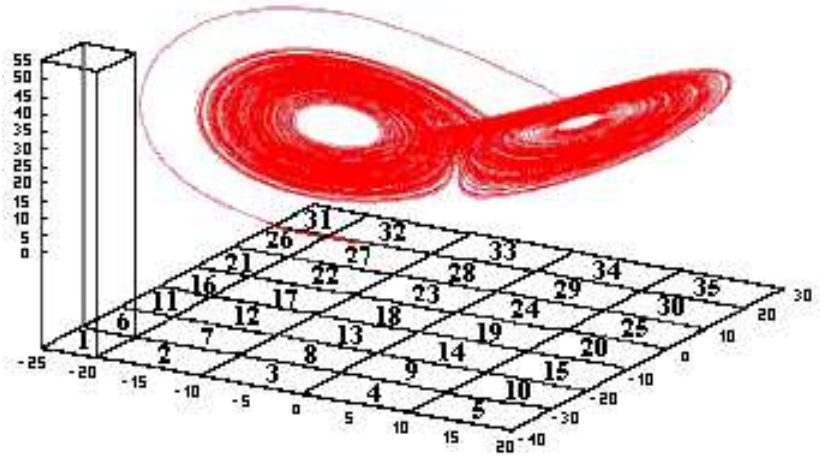

FIGURA 5.21 – Divisão do gráfico do espaço de estados do atrator de Lorenz em 35 áreas.

Através da simulação utilizando as Redes Neurais por 20.000 unidades de tempo, obteve-se os dados necessários para a identificação da quantidade de pontos em cada área. Partindo-se destes dados, foram construídos histogramas para as simulações com cada uma das redes treinadas e para os dados originalmente gerados pelas equações. Um exemplo da comparação entre os histogramas construídos para as simulações com as equações e com duas das redes treinadas para se comportar como o Sistema de Lorenz, pode ser observada na Figura 5.22. As redes usadas para esta comparação são: a rede 3-11-3 para o treinamento com $\Delta t = 0,03$, e a rede 3-16-3 para o treinamento com $\Delta t = 0,06$.

Considerando todas as análises feitas, desde o erro apresentado pelas Redes Neurais treinadas, a simulação utilizando as redes e a comparação da densidade de pontos através de histogramas, identificou-se que as redes mais apropriadas para a extração de conhecimento seriam, a rede com 11 neurônios na camada oculta, treinada com dados obtidos com $\Delta t = 0,03$ e a rede com 16 neurônios na camada oculta, para os dados com $\Delta t = 0,06$. Como os métodos adotados necessitam de um número consideravelmente pequeno de neurônios na camada oculta, para fornecer um resultado satisfatório, sem elevados custos computacionais, optou-se por realizar a extração dos três formalismos a partir da rede com 11 neurônios ocultos.

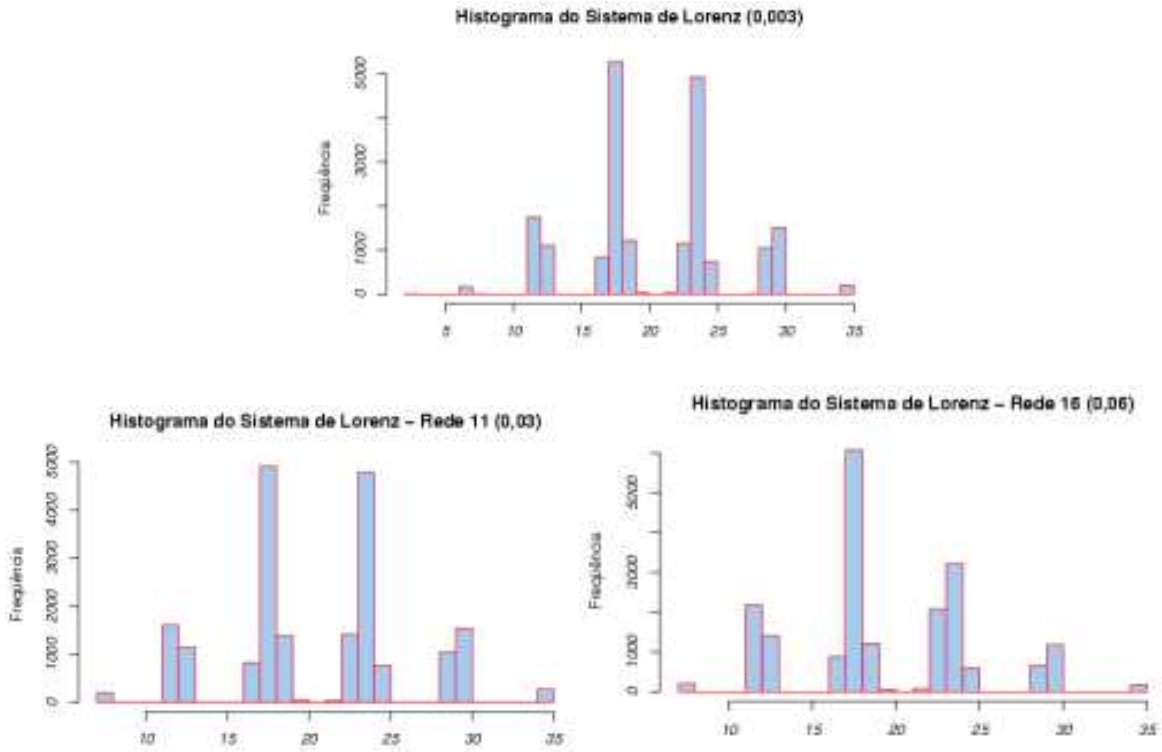

FIGURA 5.22 – Histogramas para comparação da densidade de pontos em cada área do gráfico do atrator, apresentado na Figura 5.21. O primeiro histograma (acima) foi construído utilizando os dados gerados para o treinamento das redes. O histograma do gráfico à esquerda foi construído a partir dos dados de simulação utilizando a rede com 11 neurônios na camada oculta, e o gráfico à direita, a partir da rede com 16 neurônios ocultos.

Após todos os experimentos executados para a rede selecionada e a análise dos mesmos, novas redes foram treinadas utilizando a totalidade dos dados gerados originalmente, desta forma, considerando a constante de tempo $\Delta t = 0,003$.

A escolha de novos experimentos baseados em todos os dados gerados para o sistema de Lorenz, deve-se ao número de estados e transições apresentadas usando a rede com 11 neurônios na camada oculta. Através do histograma construído a partir da quantidade de dados que pertenciam a cada estado, pode-se observar que os 10 estados selecionados inicialmente só cobriam 55% dos dados de treinamento, e para cobrir a grande maioria dos dados seria necessário, no mínimo, 30 funções de pertinência representativas dos estados difusos. Embora o AFDif extraído desta rede tenha apresentado várias características representativas do comportamento do sistema, também foram obtidas inúmeras transições que não correspondiam claramente ao sistema. Os resultados obtidos para a extração do AFDif, com a rede com 11 neurônios ocultos, são apresentados no Apêndice D. Estão anexados o histograma utilizado para a identificação do número de estados para o AFDif, o autômato extraído com 10 estados e o gráfico em três dimensões com os centróides respectivos aos 10 estados.

Para os novos experimentos foram testadas várias topologias de rede, até que fosse encontrada a rede que representasse o comportamento demostrados pelo sistema. Os erros para as rede treinadas são mostrados na Tabela 5.7.

Topologia da RNR	$\Delta t = 0,003$ Erro Médio Quadrático
3-2-3	0,37
3-3-3	0,12
3-4-3	0,08
3-5-3	0,06

TABELA 5.7 – Erro médio quadrático de treinamento das RNR usadas para aprender o comportamento do sistema de Lorenz, com $\Delta t = 0,003$.

A topologia de rede, com menor número de neurônios na camada oculta, que conseguiu aprender o comportamento do atrator de Lorenz, através dos dados gerados com $\Delta t = 0,003$, foi a rede com 5 neurônios ocultos. A simulação utilizando esta RNR pode ser vista na Figura 5.23, e o histograma para comparação da densidade de pontos em cada área do gráfico do atrator é mostrado na Figura 5.24.

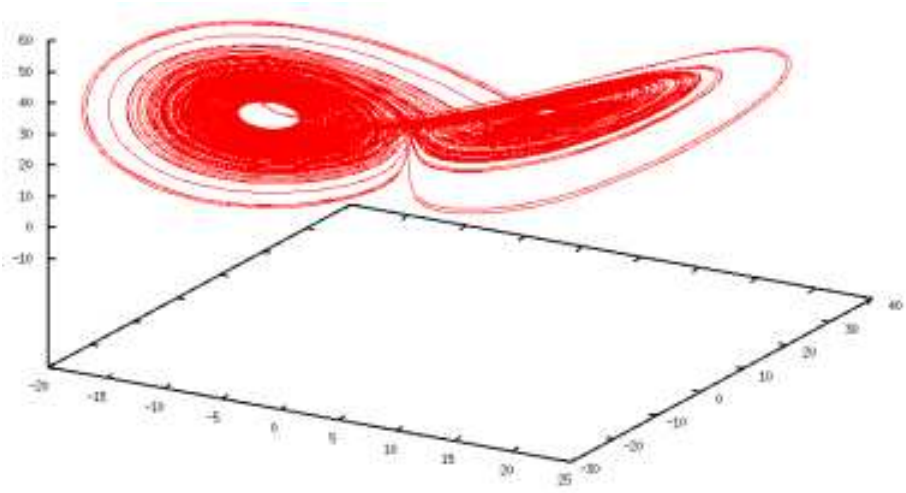

FIGURA 5.23 – Representação do atrator de Lorenz utilizando a RNR 3-5-3. O gráfico apresenta o atrator contruído através da simulação utilizando a rede com $\Delta t = 0,003$.

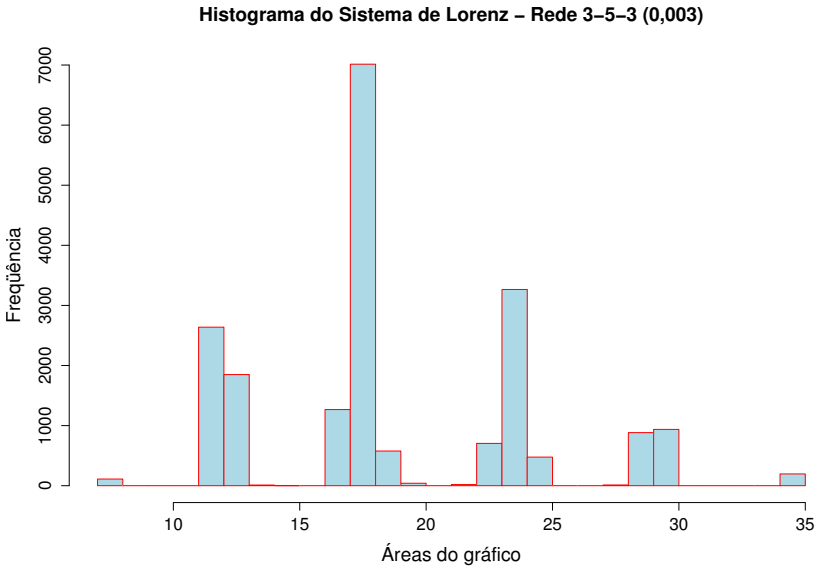

FIGURA 5.24 – Histograma da densidade de pontos em cada área do gráfico do atrator, construído utilizando a RNR com 5 neurônios na camada oculta, para a obtenção dos dados de simulação.

Os resultados obtidos com a utilização da RNR 3-5-3, com $\Delta t = 0,003$, para a aplicação dos métodos de extração serão apresentados a seguir.

Extração do Autômato Finito Difuso

Para a extração do autômato finito difuso para o sistema de Lorenz, usando a rede com 5 neurônios na camada oculta, foram construídas três funções de pertinência para cada neurônio, e através das combinação destas funções, identificou-se as funções de pertinência para cada estado do autômato. Verificou-se que, das funções de pertinência criadas, 16 eram utilizadas para representar os dados de treinamento.

A definição de quantos estados seriam utilizados, foi feita através da construção do histograma mostrado na Figura 5.25, para o qual foram calculados todas as funções de pertinência dos estados para todos os dados de treinamento. A partir da identificação de qual função possui maior grau de pertinência para cada dado, pode-se construir o histograma, onde foi possível identificar que os seis primeiros estados representam 90% dos dados amostrados, desta forma, optou-se por utilizar os seis estados para a identificação das transições e representação do sistema através do autômato.

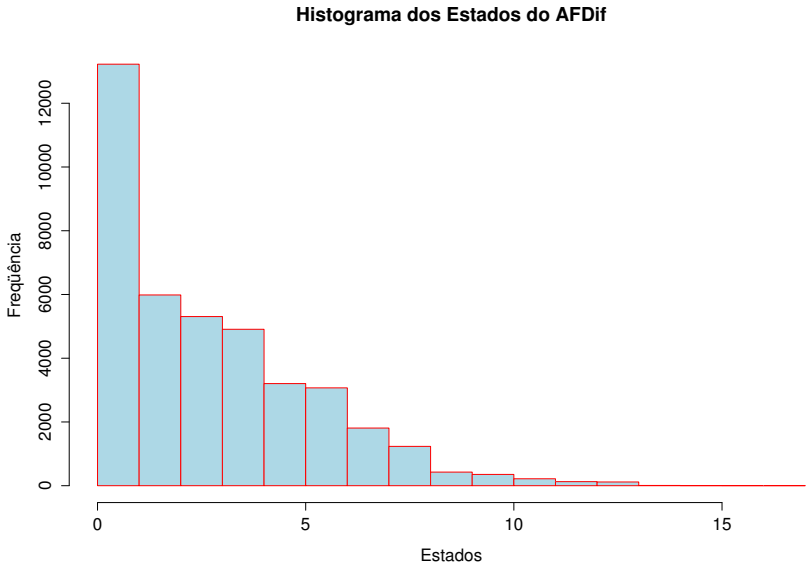

FIGURA 5.25 – Histograma da densidade dos dados representados por cada um dos estados extraídos para o AFDif. O eixo x representa os estados e o eixo y, a quantidade de dados representada por cada um dos estados.

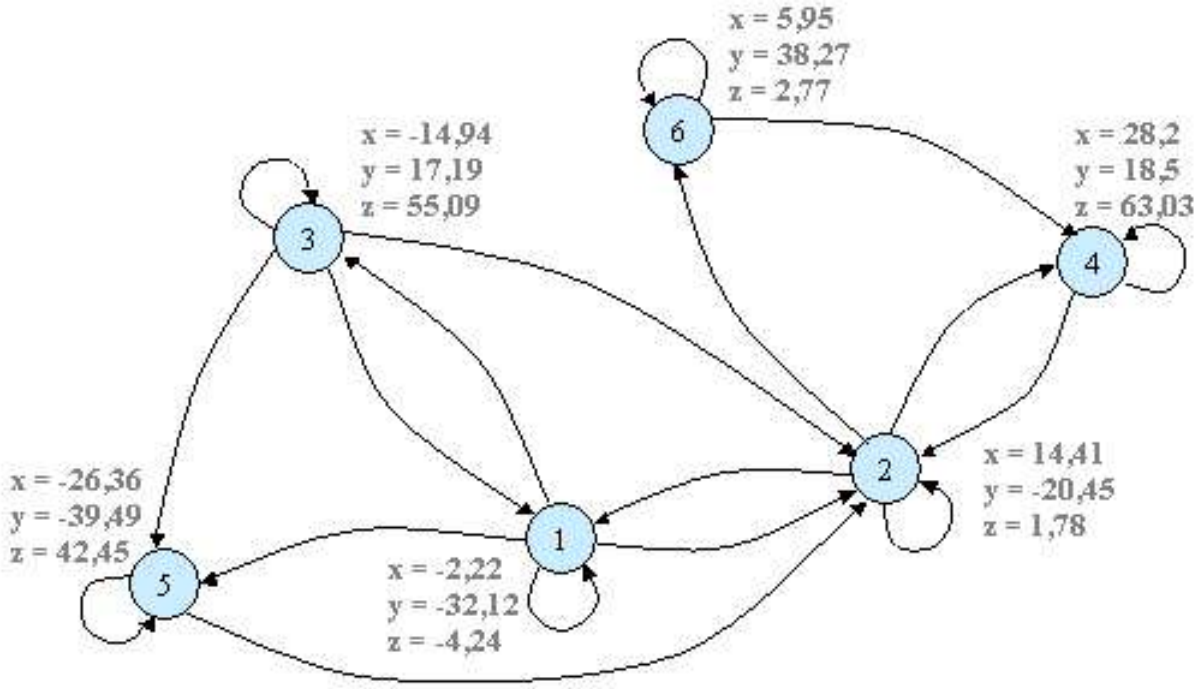

FIGURA 5.26 – Autômato finito difuso extraído para o sistema de Lorenz, apartir da RNR com 5 neurônios na camada oculta.

A Figura 5.26 apresenta o AFDif extraído para o sistema através da utilização da Rede Neural com 5 neurônios na camada oculta. Na figura podem ser observados os 6 estados extraídos e as transições encontradas entre eles. Os valores aproximados dos centróides são mostrados na figura associados aos seus respectivos estados, e também podem ser visualizados em um espaço tri-dimensional, cujas coordenadas correspondem às variáveis x, y e z, como pode ser visto na Figura 5.27. Quando da observação da Figura 5.27, deve-se considerar que a extração do conhecimento e representação do mesmo em forma de autômato foi baseada na RNR treinada com séries temporais identificadas dentro do atrator e também por séries que iniciavam por pontos transientes, possibilitando, desta forma, que os centróides identificados estejam posicionados fora do atrator. O AFDif, mostrado na Figura 5.26, está representado em forma de grafo, tendo sido construído a partir dos pontos determinados por cada centróide em um gráfico de duas dimensões (x e y), para tornar a compreensão do sistema, em forma de autômato, mais intuitiva.

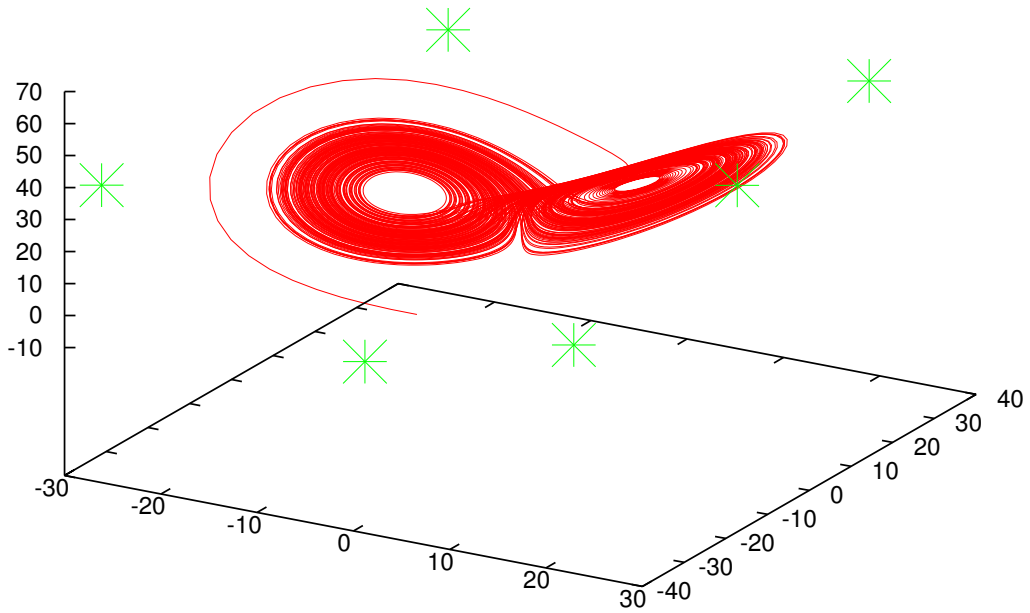

FIGURA 5.27 – Representação do atrator de Lorenz com os centróides dos 6 estados referentes ao AFDif (cor verde), com dimenções referentes as variáveis x, y e z do sistema de Lorenz.

Analisando o AFDif extraído, podem ser observadas transições que representam o gráfico característico do atrator de Lorenz, por exemplo, observa-se a possibilidade do sistema permanecer passando entre os estados 2, 6 e 4, sendo que, a distinção entre os dois lados do gráfico pode ser observada através dos estados 1, 3 e 5 no lado esquerdo do autômato, e 2, 4 e 6 no lado direito do autômato. Os estados do AFDif da Figura 5.26 podem ser descritos através das matrizes de transição apresentadas na Tabela 5.8, considerando a Eq. 5.14. Para o cálculo das matrizes utilizou-se como referência, os valores dos centróides de cada estado para as variáveis $x(t)$, $y(t)$ e $z(t)$.

$$\begin{bmatrix} x(t+1) \\ y(t+1) \\ z(t+1) \end{bmatrix} = A * \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} + B \tag{5.14}$$

Estado	Matriz A	Matriz B
1	$\begin{bmatrix} 0,916959 & 0,094477 & -0,000191 \\ 0,24643 & 0,983213 & -0,004196 \\ -0,243807 & -0,116272 & 1,00958 \end{bmatrix}$	$\begin{bmatrix} 2,86972 \\ 0,118094 \\ -4,2318 \end{bmatrix}$
2	$\begin{bmatrix} 0,908354 & 0,094602 & -0,001717 \\ 0,072668 & 0,98951 & -0,037575 \\ 0,059413 & 0,034033 & 0,967861 \end{bmatrix}$	$\begin{bmatrix} 2,21278 \\ 0,0000178 \\ -1,00147 \end{bmatrix}$
3	$\begin{bmatrix} 0,895194 & 0,094507 & 0,006009 \\ -0,197406 & 0,985695 & 0,123045 \\ -0,004959 & -0,038909 & 0,915299 \end{bmatrix}$	$\begin{bmatrix} -2,55795 \\ -8,32019 \\ 4,05041 \end{bmatrix}$
4	$\begin{bmatrix} 0,894907 & 0,093458 & -0,009348 \\ -0,21934 & 0,949552 & 0,192579 \\ 0,28991 & 0,238795 & 0,943915 \end{bmatrix}$	$\begin{bmatrix} 1,53291 \\ 16,238 \\ -7,99781 \end{bmatrix}$
5	$\begin{bmatrix} 0,898722 & 0,093667 & 0,009348 \\ -0,136563 & 0,96528 & 0,192675 \\ -0,254816 & -0,18657 & 0,938955 \end{bmatrix}$	$\begin{bmatrix} 0,238234 \\ -11,8894 \\ -8,71056 \end{bmatrix}$
6	$\begin{bmatrix} 0,914267 & 0,094048 & -0,001049 \\ 0,185772 & 0,972148 & -0,022507 \\ 0,295728 & 0,1405 & 1,00298 \end{bmatrix}$	$\begin{bmatrix} -2,11655 \\ 0,357893 \\ -6,48534 \end{bmatrix}$

TABELA 5.8 – Matrizes de transição dos estados do AFDif extraído para o sistema de Lorenz.

O AFDif pode ser visualizado também, através do gráfico contendo as funções de pertinência plotada em um espaço 3D, conforme mostrado na Figura 5.28. Cada curva do gráfico representa a região ocupada por um estado, podendo-se observar os locais onde os estados possuem maior grau de pertinência.

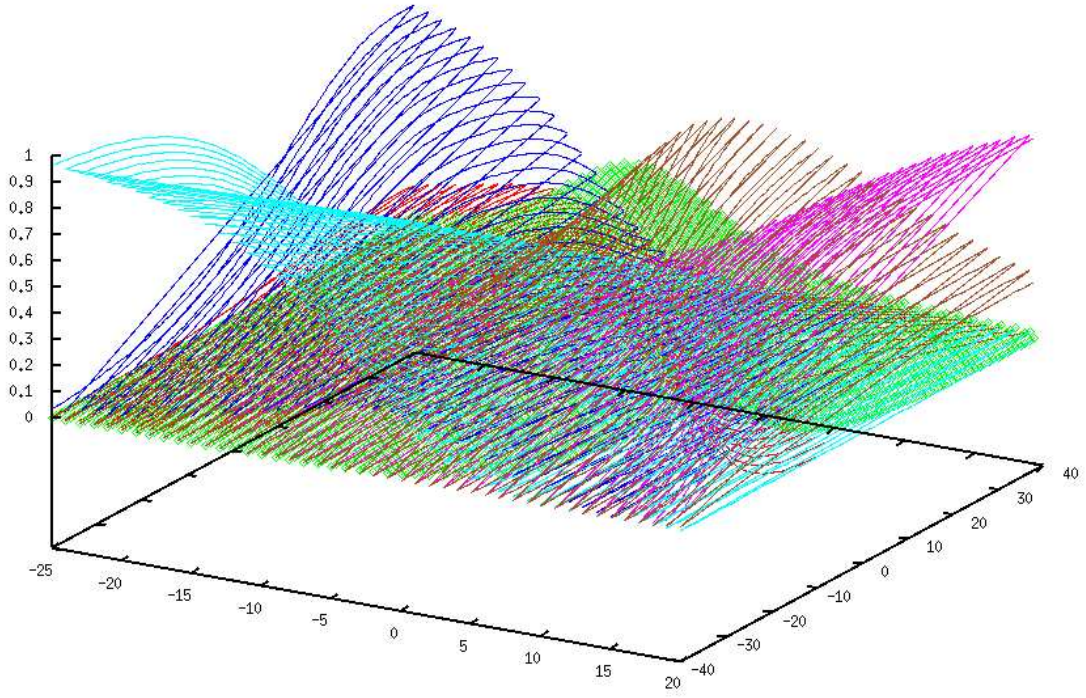

FIGURA 5.28 – Gráfico das curvas produzidas através das funções de pertinência de cada estado.

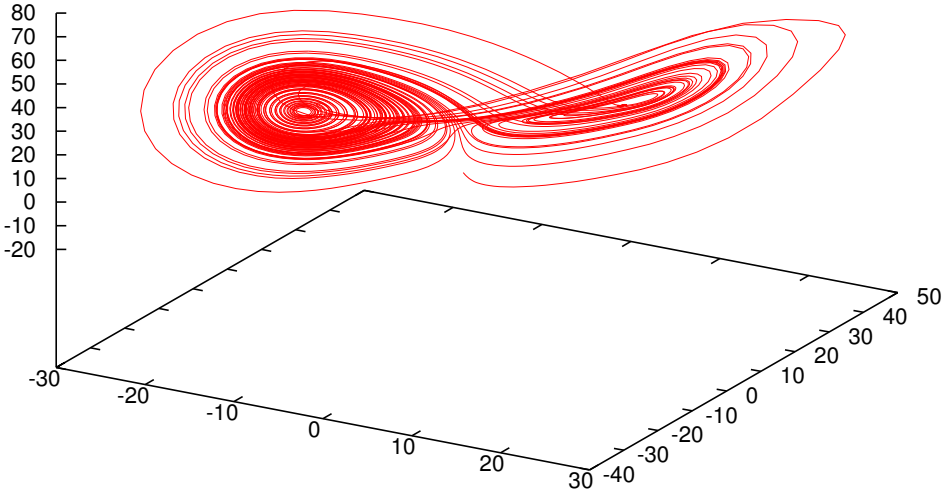

FIGURA 5.29 – Representação do atrator de Lorenz utilizando o AFDif extraído a prtir da RNR 3-5-3. O gráfico apresenta o atrator contruído através da simulação utilizando o autômato com $\Delta t = 0,003$.

Através da simulação do sistema utilizando o AFDif extraído, pode-se constatar que o mesmo consegue representar o comportamento identificado no atrator de Lorenz durante um certo período de tempo. A visualização final da simulação do sistema com o AFDif, pode ser vista na Figura 5.29. Observam-se nos gráficos da Figura 5.30, os valores referentes às variáveis de estado x, y e z, durante a simulação executada.

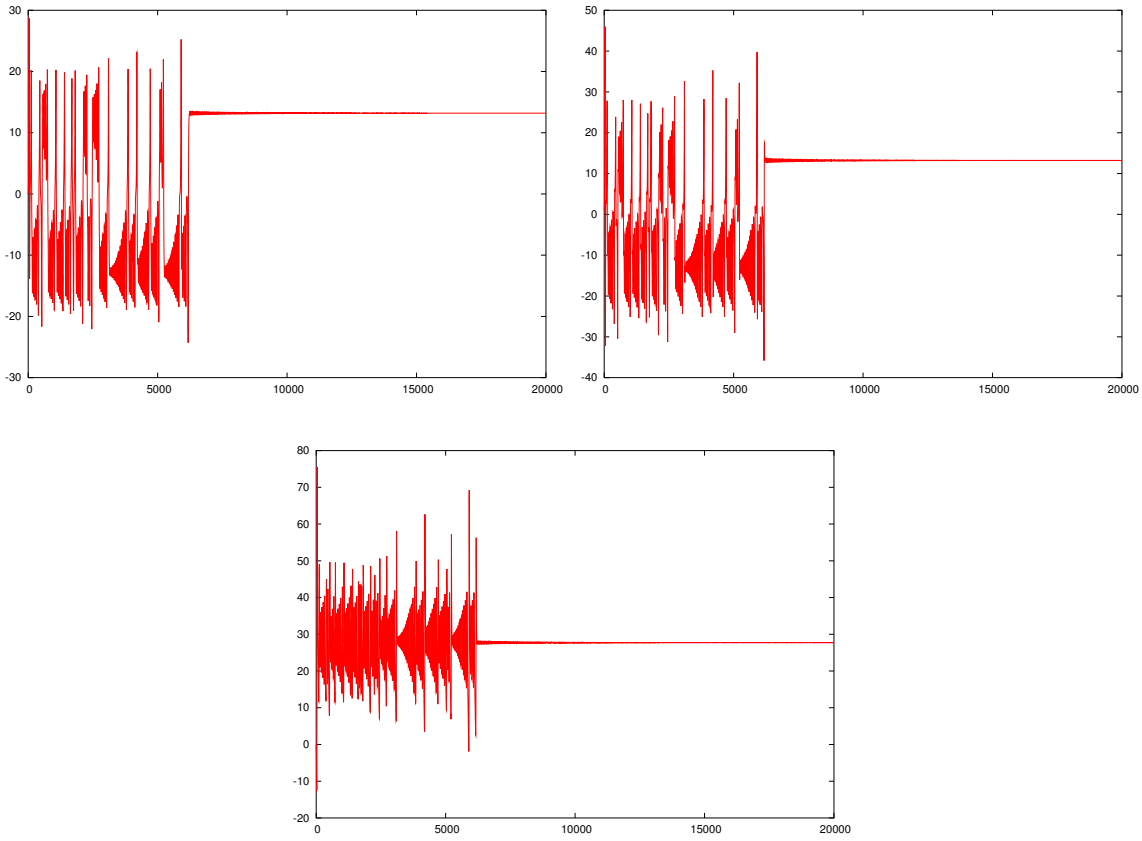

FIGURA 5.30 – Valores de x, y e z plotados através do tempo, gerados utilizando o AFDif extraído a partir da RNR com 5 neurônios ocultos. Simulação executada durante um intervalo de 20.000 unidades de tempo, com condições iniciais $x(t) = 0$, $y(t) = 0$ e $z(t) = 0$.

Extração da Cadeia de Markov

Para a extração da cadeia de Markov utilizou-se o algoritmo de clusterização *K-means*, sendo esta clusterização efetuada no espaço neural, utilizando-se todos os dados de treinamento. As ativações dos neurônios da camada oculta foram apresentados ao *K-means*, para a clusterização e, a partir dos clusters formados, plotou-se todos os dados referentes a cada cluster no espaço de ativação da rede. No gráfico da Figura 5.31 observa-se os 6 *clusters* formados, cada um representado por uma cor diferente. Na figura os *clusters* são apresentados com todas as possíveis

combinações entre as 5 ativações usadas para a clusterização. A escolha do número de *clusters* selecionados para a utilização do *K-means* se deve a análise efetuada para a extração do AFDif, visto que, 6 estados foram suficientes para a representação da grande maioria dos dados.

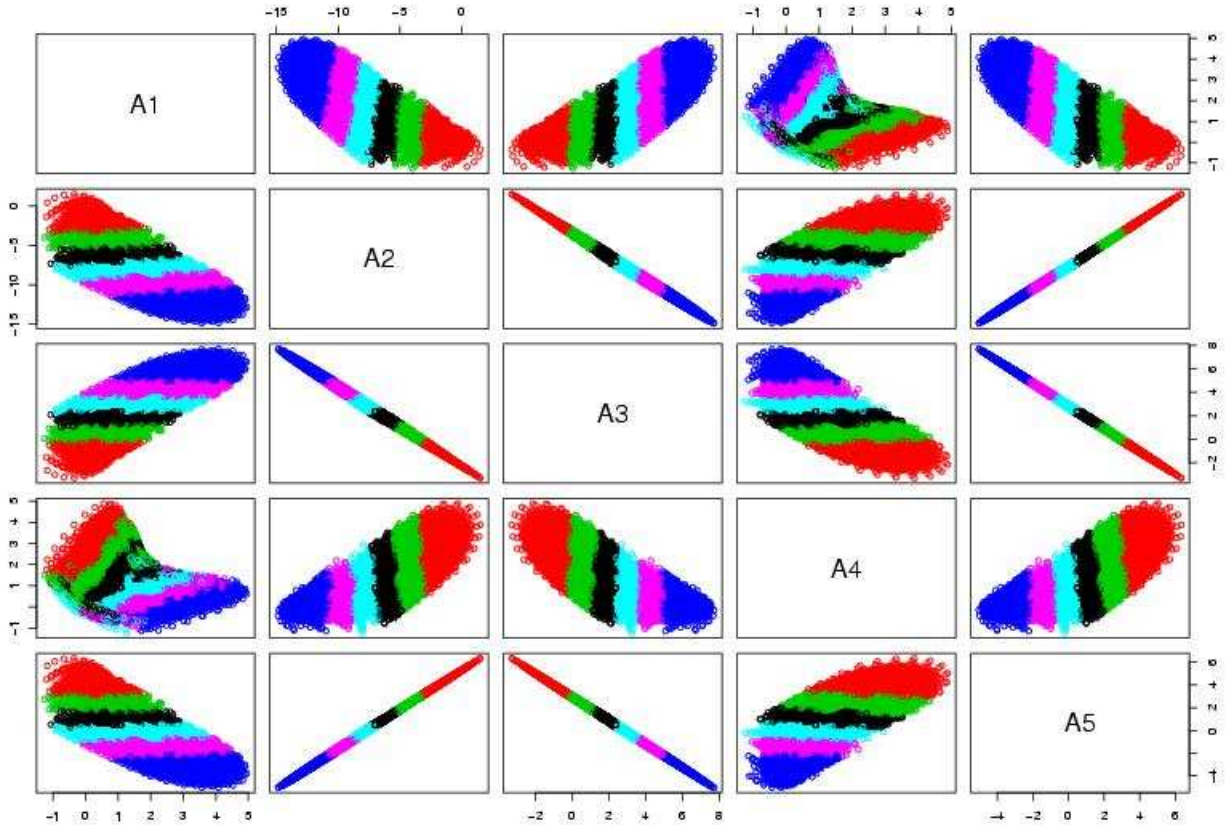

FIGURA 5.31 – Visualização dos 6 *clusters* formados através do *K-means*, e plotados com todas as combinações das variáveis de entrada do sistema de Lorenz.

A Figura 5.32 apresenta a cadeia de Markov extraída para o sistema através da utilização da Rede Neural com 5 neurônios na camada oculta. Na figura podem ser observados os 6 estados extraídos e as transições encontradas entre os estados. Associados às transições, encontram-se os valores de probabilidade destas ocorrerem. A probabilidade de cada transição ocorrer foi calculada através das freqüências encontradas para cada transição da cadeia a partir das séries temporais utilizadas para a clusterização.

A cadeia de Markov (Figura 5.32) está representada em forma de grafo, sendo que seus estados estão localizados nos pontos aproximados para cada centróide em um gráfico de duas dimensão, onde se considera somente os valores de x e y. Este modelo apresenta um número maior de transições que o encontrado no AFDif, embora as posições dos centróides relativos aos estados apresentem bastante semelhança. O comportamento, aqui demostrado através das

transições apresenta a possibilidade de ocorrerem transições entre os estados 4, 5 e 6, e entre os estados 1, 2 e 3 em ambas as direções, característica esta, apresentada pelo sistema de Lorenz.

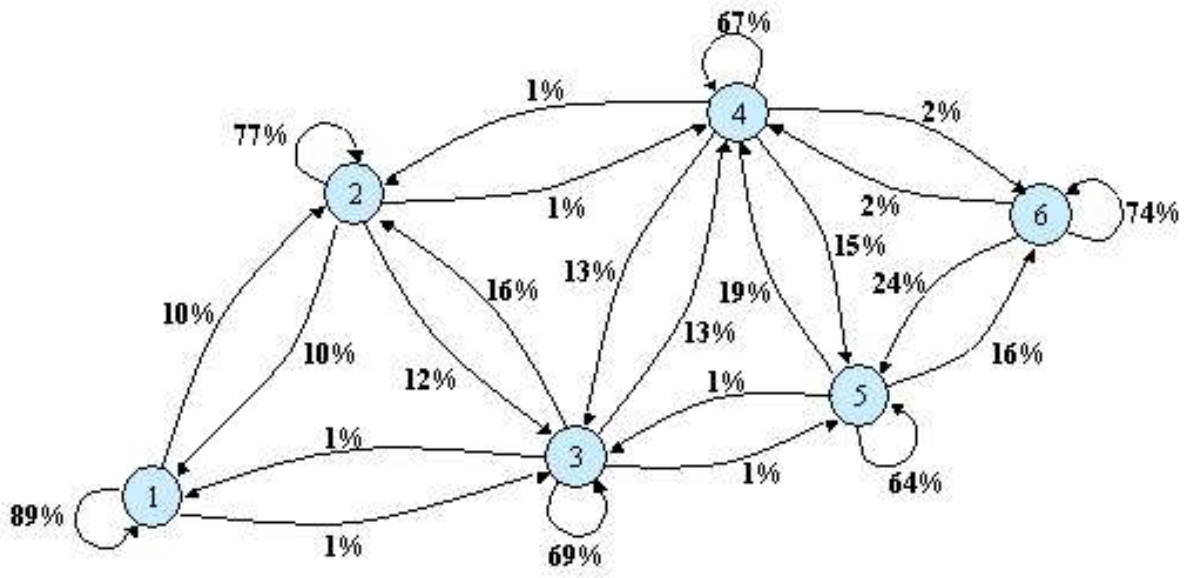

FIGURA 5.32 – Cadeia de Markov extraída para o sistema de Lorenz, a partir da RNR com 5 neurônios na camada oculta.

Extração do Autômato Finito Determinístico

Para a construção e interpretação do sistema de Lorenz através de um autômato finito determinístico são encontradas inúmeras dificuldades. Considerando que cada um dos clusters identificados através do *K-means* ocupam uma área bastante grande do espaço de estados do sistema, seriam necessários muitos estados para que se conseguisse determinar todas as transições que representassem o comportamento desejado. Uma tentativa de construção do AFDet, baseado na mesma clusterização efetuada para a extração da cadeia de Markov, pode ser visto na Figura 5.33. Os mesmos seis estados são identificados nos dois autômatos apresentados na figura. As transições foram construídas baseando-se em uma abstração da cadeia de Markov da Figura 5.32, sendo apresentadas no autômato, somente as transições entre estados que possuíam os maiores valores de probabilidade.

Através da figura, podem ser identificado alguns dos comportamentos apresentados pelo sistema de Lorenz. Por exemplo, a possibilidade do sistema permanecer nas regiões determinadas pelos estados 2 e 3, e pelos estados 4 e 5.

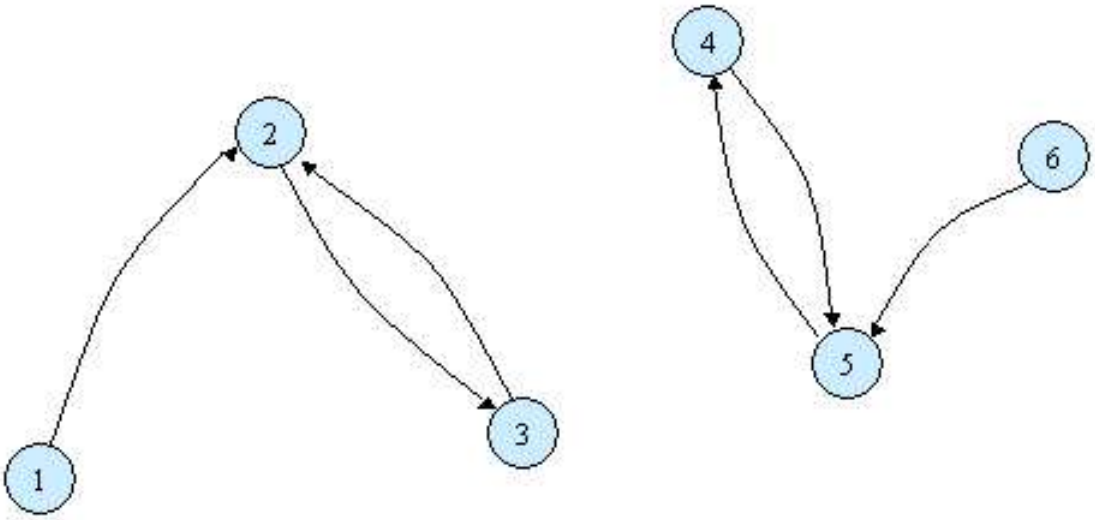

FIGURA 5.33 – Autômatos finitos determinísticos extraídos para o sistema de Lorenz, a partir da RNR com 5 neurônios na camada oculta.

Capítulo 6

Conclusão

O estudo sobre extração de conhecimento origina-se da dificuldade de interpretar o conhecimento armazenado dentro de uma RNA. Devido ao bom desempenho geralmente demonstrado pelas RNAs, um grande número de aplicações tem feito uso desta técnica. No entanto, para muitas aplicações, não é importante apenas obter o conhecimento, mas também se ter a facilidade de compreendê-lo. Através da representação deste conhecimento em um modelo simbólico bem estruturado, é possível analisar o comportamento apresentado pelo sistema, e até alterá-lo para que se comporte como desejado.

Este trabalho apresentou uma abordagem para o problema da extração de conhecimento a partir de uma RNRs. O método de extração proposto é baseado na lógica difusa e no algoritmo desenvolvido por Cechin [10] para Redes Neurais não recorrentes. Para a representação do conhecimento extraído utilizou-se três formalismos que proporcionam a análise e descrição de um sistema, AFDif, cadeias de Markov e AFDet.

As aplicações tratadas no trabalho são séries temporais calculadas através de simulações da dinâmica dos sistemas abordados. Para o sistema do Pêndulo Inverso, a seqüência de posições que o pêndulo assume ao longo de possíveis trajetórias são utilizadas para o treinamento de uma RNR, e para o sistema de Lorenz, são utilizadas séries temporais geradas a partir das três equações do sistema, cobrindo, da melhor forma possível, o seu espaço de estados.

Através dos experimentos realizados, observou-se que a representação dos estados do autômato como estados difusos pode ser extraída e o grau de pertinência do estado do sistema à um estado do autômato finito pode ser interpretada como uma medida de proximidade daquele em relação ao estado do autômato.

A representação do conhecimento apresentado pelo sistema do Pêndulo Inverso na forma de um AFDif, mostrou inúmeras vantagens em relação a representação através de AFDet e cadeias de Markov. Uma das principais vantagens, foi que, com a representação de poucos estados do AFDif, foi possível identificar comportamentos nitidamente similares ao sistema original, e replicá-lo através de simulação utilizando o próprio modelo simbólico. Embora para este tipo de extração fosse necessário um gasto maior em processamento e análise das RNRs, para se obter o autômato desejado, sem dúvida, este foi o melhor modelo de representação de conhecimento obtido no trabalho.

A extração do conhecimento utilizando o algoritmo de clusterização *K-means* demonstrou que são necessários muitos estados para que fosse possível obter uma representação mais clara do sistema do Pêndulo. Levando em consideração os resultados obtidos, nota-se claramente que para um AFDet poder ser extraído, e além disso, conseguir proporcionar uma visualização clara do comportamento do sistema estudado, é necessário considerar o estado não só como um vetor no espaço de estados, mas considerá-lo, representando um região maior deste espaço. Desta forma seria possível obter autômatos com um número menor de estados, mas que representassem o comportamento do sistema com maior precisão.

Na extração de conhecimento de RNRs treinadas com séries temporais que representavam o comportamento do sistema de Lorenz, foram utilizados os três formalismos simbólicos abordados no trabalho. Através do AFDif extraído, conseguiu-se representar de forma satisfatória o comportamento do sistema em questão, mas, a melhor forma de representação encontrada para este sistema foi na forma de cadeia de Markov. Este formalismo tornou possível identificar de forma mais ampla o comportamento do atrator de Lorenz, visto que, as transições especificadas por ele proporcionam ao sistema, a possibilidade de existir transições entre os estados em ambas as direções, e a possibilidade de que o sistema permaneça transicionando entre os estados que caracterizam as "asas" do atrator. Para a representação do sistema de forma satisfatória em um modelo de AFDet, seria necessária a utilização de mais estados, ou a identificação de características que descrevessem de forma mais sólida cada transição encontrada.

De forma geral, em contraste com AFDet, um conjunto de estados do AFDif pode ser ocupado com variação de graus em qualquer ponto do tempo, geralmente diminuindo o tamanho do modelo, e tornando o sistema dinâmico inicialmente modelado mais acessível para a interpretação.

Um ponto importante identificado durante a fase de análise dos resultados, foi a constatação de que através de um único ponto, definido como protótipo, que caracteriza toda uma região do espaço de estados, é possível analisar o sistema de forma global, embora utilizando-se uma descrição pontual para cada região. Este ponto pode ser determinado através da clusterização das variáveis do sistema no espaço neural e após a identificação do protótipo que represente cada região, sua transformação para o espaço de entrada do sistema propiciando a análise e compreensão de cada um dos pontos de forma simplificada.

A análise dos modelos linearizados dos sistemas, através do conceito de estabilidade de Lyapunov, permite caracterizar o sistema em seu espaço de estados, possibilitando que diversas técnicas possam ser aplicadas para a obtenção de um controlador para o sistema dinâmico. Através dos coeficientes de Lyapunov, calculados para cada região do espaço de estados, é possível formular um conceito para cada região. Utilizando a média dos coeficientes de cada região, ou estado, como é determinado em autômatos finitos, pode-se explicar o comportamento apresentado pelo sistema em cada região onde ele pode ser encontrado.

Matrizes que caracterizam cada uma das regiões podem ser obtidas, através da utilização dos protótipos representantes de cada região. Desta forma, obtém-se uma ferramenta importante para a análise do sistema em questão e para o projeto de um controlador que considere o sistema de forma diferenciada em cada uma das suas regiões do espaço de trabalho, obtendo assim, um controlador melhor e específico para a sua aplicação. A análise de estabilidade e controlabilidade

dos sistemas não lineares estudados, através de seu modelo linearizado é um ponto importante identificado nesta dissertação, podendo ser abordado em trabalhos futuras

Outro item que pode-se citar como trabalhos futuros, é a construção de um modelo que identifique automaticamente, a partir da função de ativação de cada um dos neurônios da camada oculta, qual o melhor número de funções de pertinência a serem utilizadas em cada neurônio. Até o momento esta identificação precisa ser feita através da análise humana, sem que exista um critério que determine qual o número ideal de funções de pertinênica. Uma possibilidade seria através da identificação das regiões onde a função do neurônio oculto está sendo ativada, possibilitando que, a partir da extensão desta ativação, e principalmente de sua localização, fosse determinada a utilização de 1, 2, ou 3 funções de pertinência.

Bibliografia

[1] Abarbanel, H. D. I., *Analysis of Observed Chaotic Data.* 1st edition, Springer, 1996.

[2] Abreu, G. L. C. M., Teixeira, R. L. e Ribeiro, J. F., A Neural Network-Based Direct Inverse Control for Active Control of Vibrations of Mechanical Systems. *Proceedings of the VI Brazilian Symposium on Neural Networks,* 2000.

[3] Anastacio, T. J., A recurrent neural network model of velocity storage in the vestibulo-ocular reflex. *Advances in Neural Information Processing Systems 3*, Morgan Kaufmann, 1991.

[4] Andrews, R., Dietrich, J. e Tickle, A. B., *A Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks*, Technical report, Neurocomputing Research Centre, Australia, 1995.

[5] Barto, A. G., Sutton, R. S. e Anderson C. W., Neuronlike adaptative elements that can solve difficult learning control problems. *IEEE Trans. On Systems, Man and Cybernetics*, 13(5):834-846, 1983.

[6] Battiti, R., *First and second-order methods for learning: between steepest descent and Newton's method.* Technical report, University of Trento, 1991.

[7] Bengio, Y., Frasconi, P., Gori, M. e Soda, G., Recurrent Neural Networks for Adaptive Temporal Processing. *Proceedings of the 6th Italian Workshop on Parallel Architectures and Neural Networks,* 1993.

[8] Blanco, A., Delgado, M. e Pegalajar, M. C. A method to induce fuzzy automata using neural networks. *IFSA World Congress and 20th NAFIPS International Conference*, Joint 9th. 5:3054 -3058,2001.

[9] Casey, M., The dynamics of discrete-time computation, with application to recurrent neural networks and finite state machine extraction. *Neural Computation*, 8(6):1135-1178, 1999.

[10] Cechin, A. L., *The Extraction of Rules from Neural Networks.* Aachen: Shaker Verlag, Informatik, Phd. Dissertation, 1998.

[11] Cechin, A. L., Souto, A. e Oliveira, L. P. L., The interpretation of neural networks in the simulation of physical systems. *EUROSIM 2001 CONGRESS, Delft. Proceedings of the 4th International Eurosim 2001 Congress*, Delft, The Netherlands: DBSS 2001, p. 95-95, 2001.

[12] Cechin, A. L., Simon, D. R. P. e Stertz, K., State Automata Extraction from Recurrent Neural Nets using k-Means and Fuzzy Clustering. *XXIII International Conference of the Chilean Computer Science Society*, p. 73-78, 2003.

[13] Cloete, I. e Zurada, J. M., *Knowledge-Based Neurocomputing.* The MIT Press, Cambridge, Massachusetts, 2000.

[14] Clouse, D. S., Giles, C. L., Horne, B. G. e Cottrell, G. W., Time-Delay Neural Networks: Representation and Induction of Finite State Machines. *IEEE Trans. on Neural Networks*, 8(5):1065, 1997.

[15] Costa, F., Frasconi, P., Lombardo, V. e Soda, G., Learning Incremental Syntactic Structures with Recursive Neural Network. *Proceedings 4th Int. Conf. On Knowledge-Based Engineering Systems.* 2:458-461. IEEE Press, 2000.

[16] Cozzio, R. A., Approximation of Differential Equations Using Neural Networks. Em *Knowledge-Based Neurocomputing*, I.Cloete e J.M.Zurada (eds.). The MIT Press, Cambridge, Massachusetts, 2000.

[17] Craven, M. e Shavlik, J., *Rule Extraction: Where Do We Go from Here?.* citeseer.nj.nec.com/202232.html (último acesso: abril de 2003), 1999.

[18] Cybenko, G., *Continuos valued neural networrks with two hidden layer are sufficient.* Technical report, Department of Computer Science, Tufts University, 1988.

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

[20] Das, S. e Mozer, M., Dynamic On-Line Clustering and State Extraction: An Approach to Symbolic Learning. *Neural Networks*, 11(1):53-64, 1998.

[21] Devaney, R. L., *An Introduction to Chaotic Dynamical Systems.* Addison-Wesley, 1989.

[22] Duch, W., Adamczak, R. e Grabzewski, K., A New Methodology of Extraction, Optimization and Application of Crisp and Fuzzy Logical Rules. *IEEE Transactions on Neural Networks*, 12(2), 2001.

[23] Elman, J. L., *Finding Structure in Time, Cognitive Science*, vol.14, pp. 179-211, 1990.

[24] Fahlman, S. E., *The recurrent cascade-correlation architecture.* Technical Report CMU-CS-91-100. Carnegie-Mellon University, 1991.

[25] Fahlman, S. E., *An empirical study of learning speed in backpropagation networks.* Technical report, Carnegie Mellow University, 1988.

[26] Fahlman, S. E. e Lebiere, C., The cascade-correlation learning architecture. In R. P. Lippman, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 2.* Morgan Kaufmann, 1988.

[27] Frasconi, P., Gori, M., Maggini, M. e Soda, G., Unified integration of explicit knowledge and learning by example in recurrent networks. *IEEE Transactions on Knowledge and Data Engineering*, 7(2):340-346, 1995.

[28] Funuhashi, K. I., On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2:183-192, 1989.

[29] Giles, C. L., Miller, C. B., Chen, D., Chen, H. H., Sun, G. Z. e Lee, Y. C., Learning and Extracting finite-state automata with second-order recurrent neural networks. *Neural Computation*, 4(3):380. 1992.

[30] Giles, C. L., Lawrence, S. e Tsoi, A. C., Noisy Time Series Prediction using a Recurrent Neural Network and Grammatical Inference. *Machine Learning.* 2000.

[31] Grzeszczuk, R., Terzopoulos, D. e Hinton, G., NeuroAnimator: Fast Neural Network Emulation and Control of Physics-Based Models. *Proceedings of SIGGRAPH 98*, 9-20, 1998.

[32] Hagan, M. e Menhaj, M., Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6):989-993, 1994.

[33] Haykin, S., Neural Networks Expand SP's Horizons. *IEEE Signal Processing Magazine.* 13(2): 24-29, 1996.

[34] Haykin, S., *Neural Networks: A Comprehensive Foundation.* Prentice-Hall, Inc., Upper Saddle River, New Jersey, 1999.

[35] Heinen, F. J., Cechin, A., Wagner, A. e Souto, A., Simulação de Manipuladores e Modelagem Através de Redes neurais. *X SEMINCO, 2001*, Blumenau. Anais do X SEMINCO. Blumenau: Editora da FURB, 1:11-24, 2001.

[36] Hertz, J., Krogh, A. e Palmer, R. G., *Introduction to the Theory of Neural Computation*, volume Lecture Notes Volume I of Santa Fe Institute Studies in The Science of Complexity. Addison-Wesley, 1991.

[37] Hoffmann, G. A., Sobottka, M. e Oliveira L. P. L. de, Dinâmica Caótica: Uma Introdução. *Scientia*, 10(2):147-175, 2000.

[38] Hopcroft, J. E., Motwani, R., Ullman, J. D., *Introduction to Automata Theory, Languages, and Computation.* 2nd edition, Addison-Wesley, 2001.

[39] Hopfield, J. J., Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the U.S.A.*, 79:2554-2558, 1982.

[40] Horne, B. G. e Hush, D. R., Bounds on the Complexity of Recurrent Neural Network Implementations of Finite State Machines. *Neural Networks*, 9(2):243-252, 1996.

[41] Hornik, K., Stinchcombe, M. e White, H., Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359-366, 1989.

[42] Jordan, M. I., Attractor dynamics and parallelism in a connectionist sequential machine. *In Proceedings of the Eighth Conference of the Cognitice Science Society*, 1986.

[43] Kambhampati, C., Garces, F. e Warwick, K., Approximation of non-autonomous Dynamic Systems by Continuous Time Recurrent Neural Networks. *Proceedings of the IJCNN'2000*, 64-69, 2000.

[44] Khalil, H. K., *Nonlinear Systems.* 2nd. edition, Prentice Hall, 1996.

[45] Klir, G. J., *Fuzzy Set Theory: Foundations and Applications.* 11th edition, Pretice-Hall, 1997.

[46] Knuth, D. E., *The Art of Computer Programming.* Volume I: Fundamental Algorithms Addison-Wesley, 1968.

[47] Kolb, T., Ilg, W. e Wille, J., Using Time-Discrete Recurrent Neural Networks in Nonlinear Control. *Proceedings of the World Congress on Computational Intelligence '98*, 1367-1371, 1998.

[48] Kosko, B., *Neural Networks and Fuzzy Systems: a Dynamical Systems.* MIT Press, Cambridge, Massachusetts. 2000. Approach to Machine Intelligence. Prentice Hall, Inc., Englewood Cliffs, 1992.

[49] Krishnaiah, P. e Kanal, L., *Classification, Pattern Recognition, and Reduction of Dimensionality*, Vol. 2 of Handbook of Statistics, Amsterdam, North Holland, 1982.

[50] Kruse, R., Gebhardt, J. e Klawonn, F., *Fuzzy Systeme.* B. G. Teubner, Stuttgart, 1993.

[51] Lang, K. J. e Hinton, G. E., *The development of the time-delay neural networks architecture for speech recognition.* Technical Report CMU-CS-88-152. Carnegie-Mellon University, 1988.

[52] Lorenz, E. N., *The essence of chaos.* University of Washington Press, 1993.

[53] Manning, C. D. e Schutze, H., *Foundations of Statistical Natural Language Processing*, MIT Press Cambridge, Massachusetts, 4th ed., 2000.

[54] Melnik, O., Pollack, J., *Exact Representations from Feed-Forward Networks.* Technical Report CS-99-205, Brandeis University, 1999.

[55] Melnik, O., Decision Region Connectivity Analysis: A method for analyzing high-dimensional classifiers. *Machine Learning Journal*, 2001.

[56] Narendra, K. S. e Parthasarathy, K., Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1: 4-27, 1990.

[57] Netto, S. M. C., Dutra, M. S. e Evsukoff, A., Fuzzy Systems to Solve Inverse Kinematics Problem in Robots Control Applications to an Hexapod Robots's Leg. *Proceedings of the VI Brazilian Symposium on Neural Networks*, 150-155, 2000.

[58] Nguyen, H. T., A first course in fuzzy logic. 2nd edition, Chapman&Hall, 2000.

[59] Noord, G. J. M. van, *FSA Utilities: A Toolbox to Manipulate Finite-state Automata.* http://odur.let.rug.nl/~vannoord/papers/fsa/node1.html (última consulta: abril de 2003), 1998.

[60] Omlin, C. W. e Giles, C. L., Extraction of Rules from Discrete-Time Recurrent Neural Networks. *Neural Networks*, 9(1): 41-52, 1996.

[61] Omlin, C. W. e Giles, C. L., Symbolic Knowledge Representation in Recurrent Neural Networks: Insights from Theoretical Models of Computation. *Knowledge-Based Neurocomputing*, I.Cloete e J.M.Zurada (eds.), 2000.

[62] Ott, E., *Chaos in Dynamical Systems.* Cambridge University Press, 1993.

[63] Oussar, Y. e Dreyfus, G., How to be a gray box: dynamic semi-physical modeling. *Neural Networks*, 14:1161-1172, 2001.

[64] Oyama, E., Agah, A., MacDorman, K. F., Maeda, T. e Tachi, S., A modular neural network architecture for inverse kinematics model learning. *Neurocomputing*, 38-40: 797-805, 2001.

[65] Pasemann, F., Pole-balancing with different evolved neurocontrollers. *Art. Neural Net. - Proc. ICANN'97*, W.Gerstner, A. Germond, A. M. Hasler e J.D. Nicoud (eds.), 1997.

[66] Pearlmutter, B., Gradient descent: second order momentum and saturation error. In J. E. Moody, S. Hanson, and R. Lippman, editors, *Advances in Neural Information Processing Systems 2*, p. 887-894. Morgan Kaufmann, 1992.

[67] Rabiner, L. R., A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, *Proceedings of the IEEE*, 77(2):257-286, 1989.

[68] Riedmiller, M., *Rprop - description and implementation details.* Technical report. University of Karlsruhe, 1994.

[69] Rumelhart, D. E., Hinton, G. E., e Williams, R. J., Learning internal representations by back-propagating errors. *Nature*, 323:533-536, 1986.

[70] Rumelhart, D. E. e McClelland, J. L., *Parallel Distributed Processing.* The MIT Press, vol. 1, 1986.

[71] Sejnowski, T. J. e Rosenberg, C. R., Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145-168, 1987.

[72] Silva, A. B. M., Portugal, M. S. e Cechin, A. L., *Redes Neurais Artificiais e Análise de Sensibilidade: Uma Aplicação à Demanda de Importação Brasileira.* Texto Para Discussão, PPG em Economia, 11:1-44, 2000.

[73] Simon, D. R. P. e Cechin, A. L., Extração de um Autômato Finito a partir de Redes Neurais Recorrentes na modelagem do Pêndulo Inverso. *Anais do XXIII Congresso da Sociedade Brasileira de Computação, IV Encontro Nacional de Inteligência Artificial*, 1:2050-2059, 2003.

[74] Simon, D. R. P., Cechin, A. L. e Stertz, K., Uma Comparação entre Métodos de Clusterização para Construção de Autômatos Finitos a partir de Redes Neurais Recorrentes. *XXIX Conferência Latino-americana de Informática*, 2003.

[75] SNNS. *Stutgart Neural Network Simulator V4.2.* www-ra.informatik.uni-tuebingen.de/SNN. Consultado em 10/01/2003.

[76] Sun, R. e Giles, L., *Sequence Learning: Paradigms, Algorithms, and Applications.* Springer-Verlag, Heidelberg, Germany, 2001.

[77] Takagi, T. e Sugeno, M., Fuzzy Identification of Systems and Ist Application to Modeling and Control. *IEEE Transactions on Systems, Man, and Cybernetics*, 15:116-132, 1985.

[78] Tang, W. S., Lam, C. M. L. e Wang, J., Kinematic Control and Obstacle Avoidance for Redundant Manipulators Using a Recurrent Neural Network. *Proceedings of the ICANN'2001*, p. 907, 2001.

[79] Tickle, A. B., Andrews, R., Golea, M. e Diederich, J., The truth is in there: directions and challenges in extracting rules from trained artificial neural networks. *IEEE Trans. on Neural Networks*, 9:1057-1068, 1998.

[80] Tiño, P., Horne, B. G. e Giles, C. L., *Finite State Machines and Recurrent Neural Networks - Automata and Dynamical Systems Approaches.* Technical Report UMIACS-TR-95-1 e CS-TR-3396, University of Maryland, College Park, 1995.

[81] Tiño, P. e Köteles, M., Extracting Finite-State Representations from Recurrent Neural Networks Trained on Chaotic Symbolic Sequences. *IEEE Trans. On Neural Networks*, 10(2):284-302, 1999.

[82] Waibel, A., Modular construction of time-delay neural networks for speech recognition. *Neural Computation*, 1:39-46, 1989.

[83] Waibel, A., Hanazawa, T., Hinton, J. G., Shikano, K. e Lang K., Phoneme recognition using time-delay neural networks. *IEEE AS-SP Magazine*, 37:328-339, 1989.

[84] Wan, E. A. Temporal backpropagation for fir neural networks. *Proc. IEEE Int. Joint Conf. Neural Networks*, 1990.

[85] Werbos, P., Backpropagation through time : What it does and to do it. *Proceedings of IEEE*, 78:1550-1560, 1990.

[86] Wermter, S. e Sun, R., *Hybrid Neural Systems*. Springer Verlag, Heidelberg, New York. 2000.

[87] Williams, R. J. e Peng, J., Reinforcement learning algorithms as function optimizers. *International Joint Conference on Neural Networks*, 2:89-95, 1989.

[88] Williams, R. J. e Zipser, D., A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270-280, 1989.

[89] Zadeh, L. A. *Fuzzy Sets*, Information and Control, 8, 338-353, 1965.

Apêndice A

Transições AFDif - Pêndulo Inverso (I)

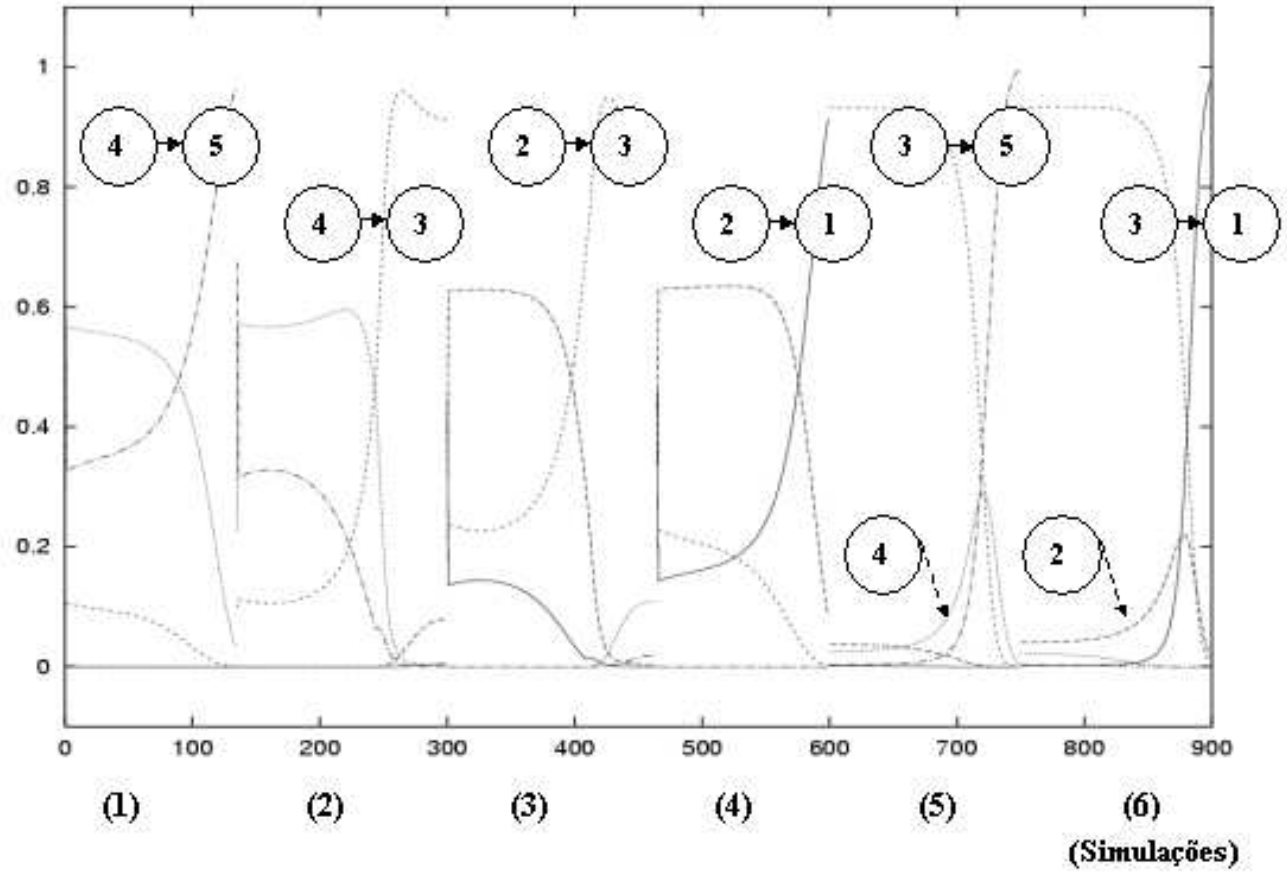

FIGURA A.1 – Gráfico de transições de estados para o sistema do Pêndulo Inverso, para as seis simulações.

Apêndice B

Cadeia de Markov - Pêndulo Inverso (II)

Força	Estado	x	ϕ	v	w
-10	1	-0,26	0,76	0,08	0,20
	2	-0,44	0,09	-0,56	0,95
	3	-0,40	-0,34	-0,24	0,13
-8	1	-0,26	0,85	-0,07	0,20
	2	-0,44	0,09	-0,44	0,84
	3	-0,40	-0,28	-0,22	0,09
-6	1	-0,28	0,89	-0,06	0,20
	2	-0,43	0,14	-0,34	0,77
	3	-0,40	-0,27	-0,18	0,05
-4	1	-0,28	0,99	-0,04	0,20
	2	-0,41	0,16	-0,23	0,71
	3	-0,40	-0,19	-0,13	0,02
-2	1	-0,27	1,1	-0,02	0,19
	2	-0,39	-0,04	-0,10	0,54
	3	-0,40	-0,09	-0,07	0,01

TABELA B.1 – Centróides dos estados das cadeias de Markov apresentadas na Figura 5.15, para valores negativos de força aplicados ao sistema.

Força	Estado	x	ϕ	v	w
0	1	-0,39	0,81	0,001	-0,03
	2	-0,55	-0,82	-0,003	0,04
	3	-0,59	-0,006	-0,002	-0,0008
2	1	-0,78	-0,005	0,10	-0,57
	2	-0,54	-1,09	0,02	-0,19
	3	-0,8	0,11	0,07	0,0006
4	1	-0,80	-0,08	0,21	-0,66
	2	-0,53	-0,99	0,03	-0,17
	3	-0,80	0,16	0,13	-0,04
6	1	-0,81	-0,10	0,33	-0,73
	2	-0,51	-0,94	0,05	-0,19
	3	-0,80	0,23	0,18	-0,06
8	1	-0,83	-0,08	0,43	-0,82
	2	-0,52	-0,85	0,07	-0,20
	3	-0,78	0,29	0,22	-0,07
10	1	-0,82	-0,06	0,54	-0,91
	2	-0,51	-0,76	0,08	-0,20
	3	-0,79	0,31	0,24	-0,14

TABELA B.2 – Valores das variáveis de estado do sistema para o centróide de cada estado, para força nula e valores positivos de força aplicados ao sistema. Estados representados nas cadeias de Markov extraídas a partir da Rede Neural com 1 neurônio na camada oculta (Figura 5.15).

Apêndice C

Extração a partir da RNR com 2 neurônios na camada oculta - Pêndulo Inverso (II)

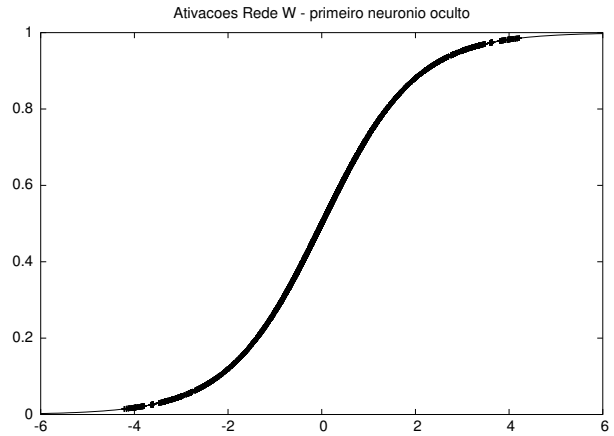

FIGURA C.1 – Funções de ativação do primeiro neurônio da camada oculta, da rede treinada para aprender o comportamento do sistema do Pêndulo Inverso.

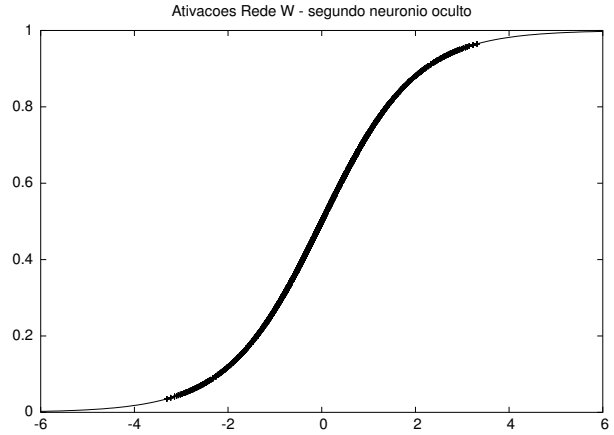

FIGURA C.2 – Funções de ativação do segundo neurônio da camada oculta, da rede treinada para aprender o comportamento do sistema do Pêndulo Inverso.

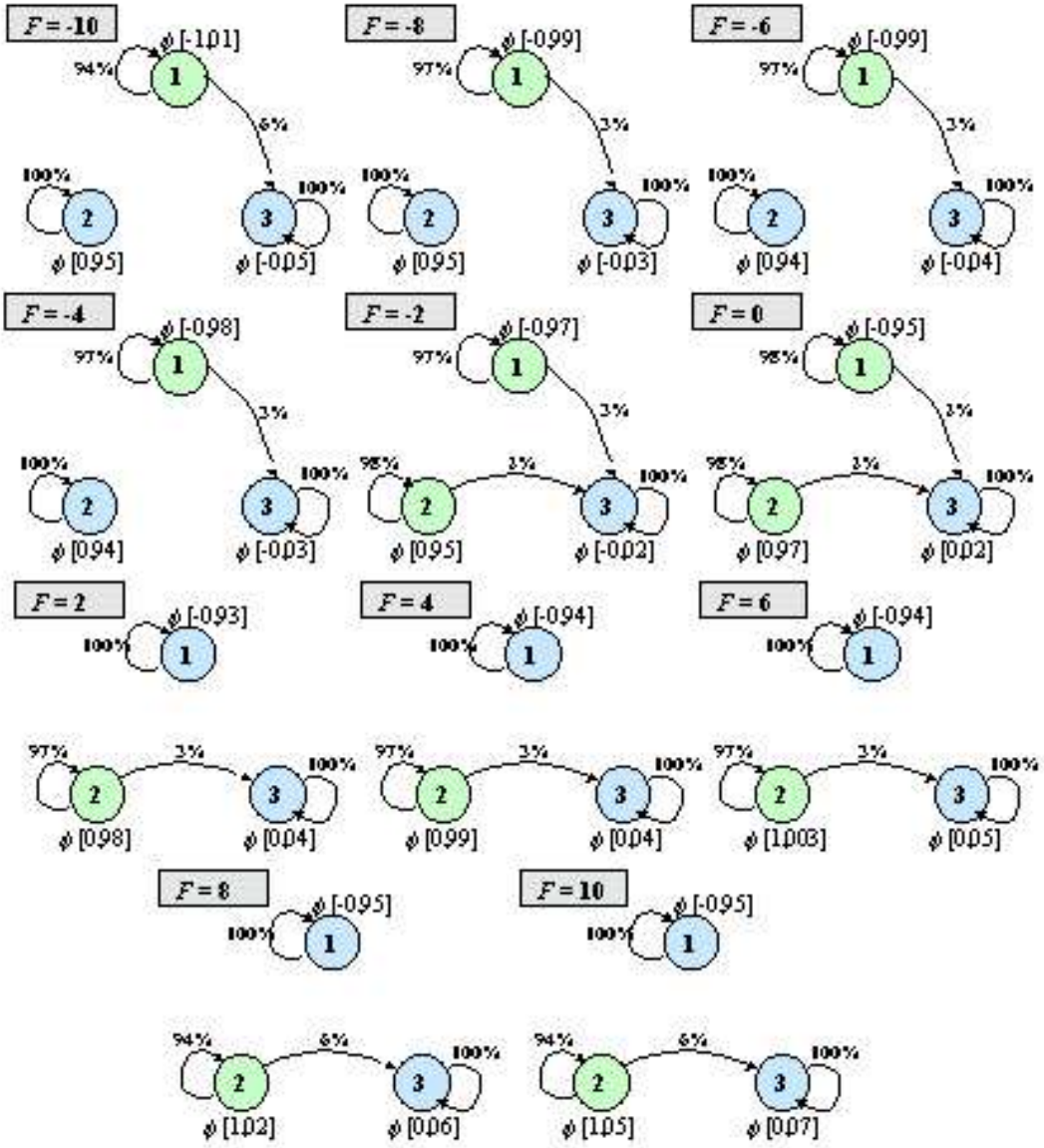

FIGURA C.3 – Cadeias de Markov extraídas a partir da RNR com 2 neurônios ocultos, para o sistema do Pêndulo Inverso, de acordo com a força aplicada ao sistema.

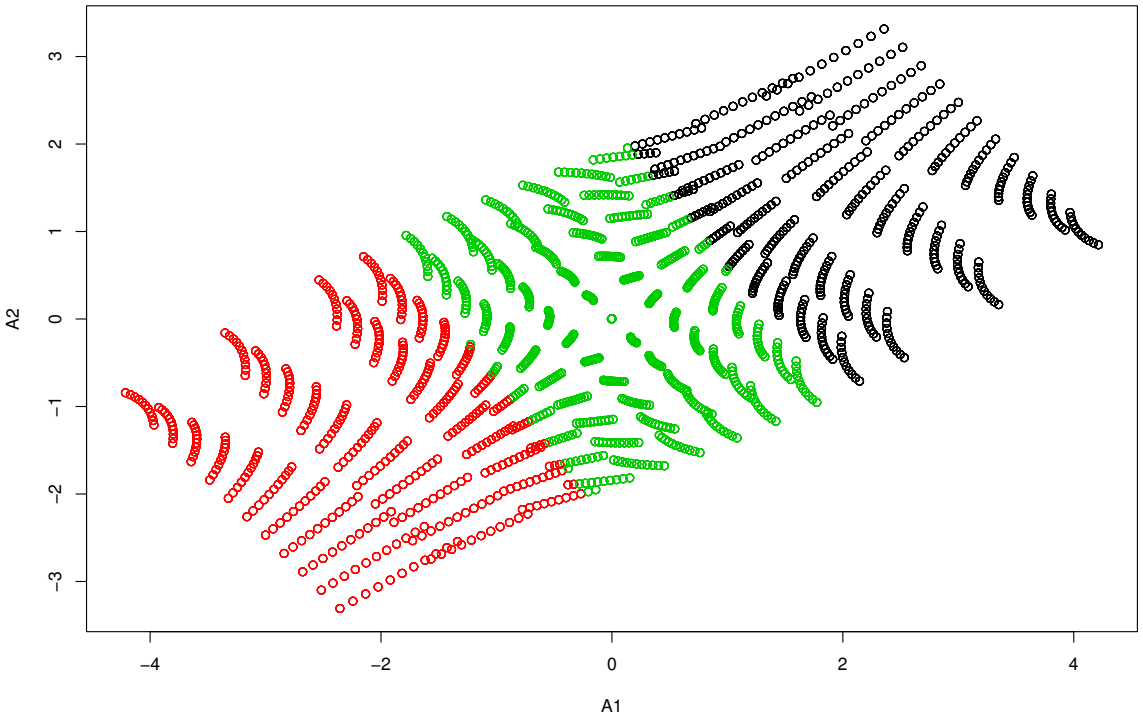

FIGURA C.4 – Clusterização com *K-means* utilizando os valores de ativação dos 2 neurônios da camada oculta. Os eixos x e y do gráfico correspondem aos valores de ativação dos neurônios.

Força	Estado	x	ϕ	v	w
-10	1	-0,39	-1,01	-0,32	-0,26
	2	-0,40	0,95	-0,37	0,97
	3	-0,41	-0,05	-0,39	0,66
-8	1	-0,39	-0,99	-0,29	-0,30
	2	-0,40	0,95	-0,30	0,89
	3	-0,40	-0,03	-0,31	0,53
-6	1	-0,39	-0,99	-0,19	-0,37
	2	-0,40	0,94	-0,23	0,81
	3	-0,40	-0,04	-0,23	0,38
-4	1	-0,39	-0,98	-0,13	-0,43
	2	-0,39	0,94	-0,15	0,73
	3	-0,40	-0,03	-0,15	0,24
-2	1	-0,39	-0,97	-0,06	-0,49
	2	-0,39	0,95	-0,08	0,64
	3	-0,39	-0,02	-0,08	0,12
0	1	-0,77	-0,95	0,006	-0,56
	2	-0,39	0,97	-0,008	0,56
	3	-0,55	0,02	-0,003	0,02
2	1	-0,78	-0,93	-0,65	-0,65
	2	-0,77	0,98	0,49	0,49
	3	-0,78	0,04	-0,09	-0,09
4	1	-0,77	-0,94	-0,73	-0,73
	2	-0,76	0,99	0,43	0,43
	3	-0,78	0,04	-0,24	-0,24
6	1	-0,77	-0,94	-0,81	-0,81
	2	-0,76	1,003	0,38	0,38
	3	-0,78	0,05	-0,38	-0,38
8	1	-0,77	-0,95	-0,89	-0,89
	2	-0,75	1,02	0,33	0,33
	3	-0,78	0,06	-0,51	-0,51
10	1	-0,77	-0,95	-0,97	-0,97
	2	-0,74	1,05	0,30	0,30
	3	-0,78	0,07	-0,64	-0,64

TABELA C.1 – Valores das variáveis de estado do sistema para o centróide de cada estados, de acordo com a força aplicada ao sistema. Estados representados nas cadeias de Markov extraídas a partir da Rede Neural com 2 neurônio na camada oculta (Figura 5.15).

Apêndice D

Extração Sistema de Lorenz - Rede 3-11-3

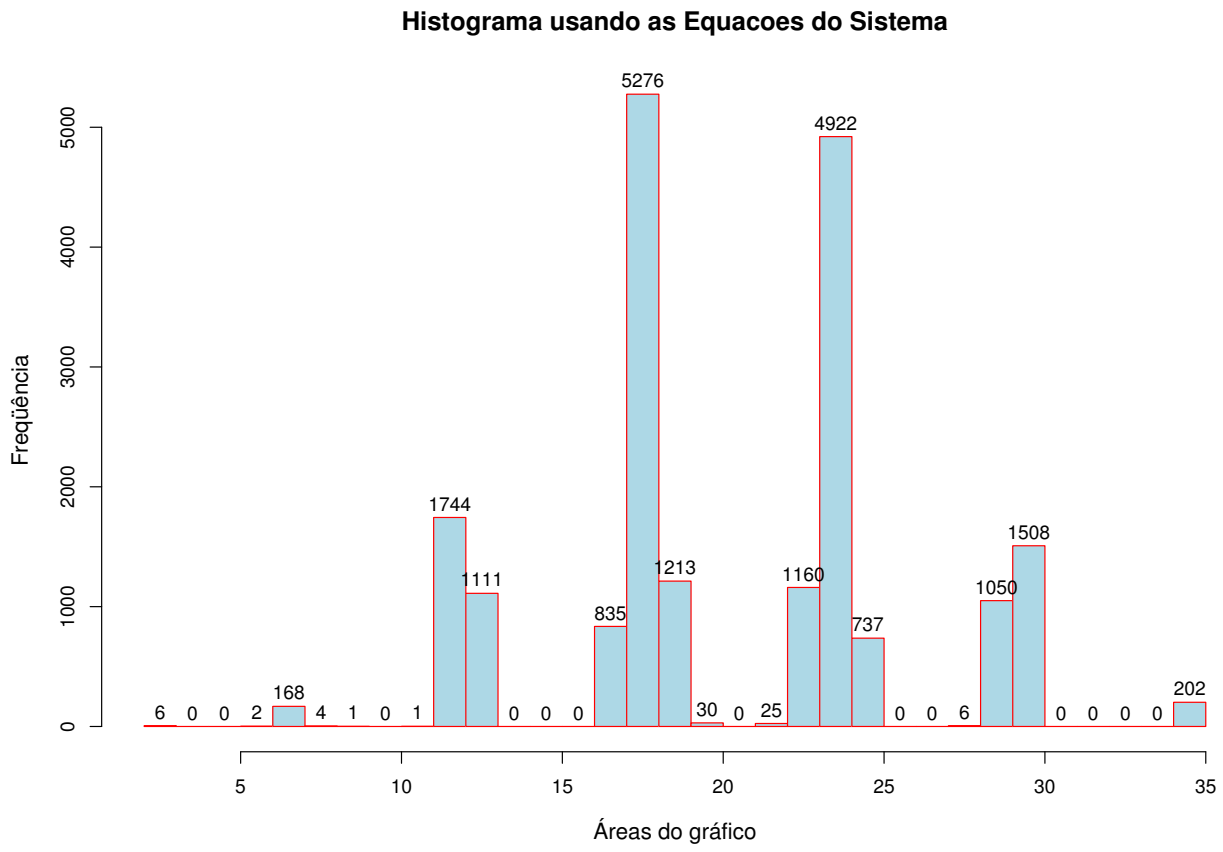

FIGURA D.1 – Histograma da densidade de pontos em cada área do gráfico do atrator, construído utilizando as Equações do sistema de Lorenz.

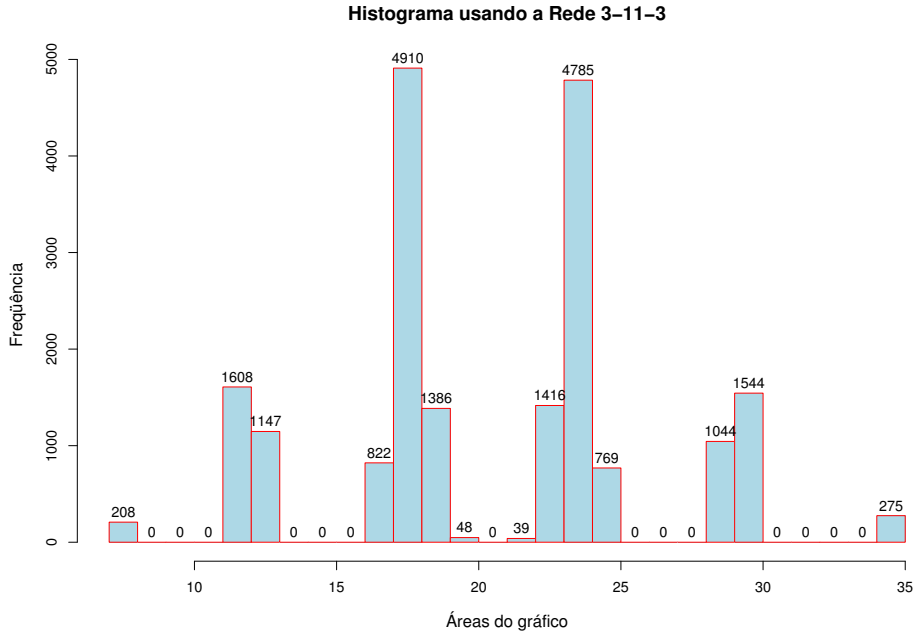

FIGURA D.2 – Histograma da densidade de pontos em cada área do gráfico do atrator, construído utilizando a RNR com 11 neurônios na camada oculta.

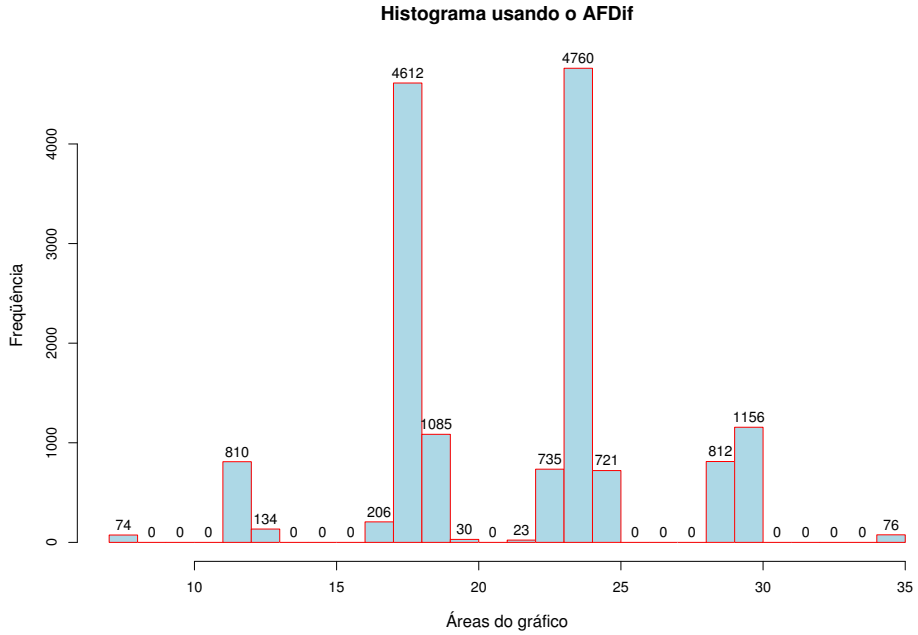

FIGURA D.3 – Histograma da densidade de pontos em cada área do gráfico do atrator, construído utilizando o AFDif extraído a partir da RNR 3-11-3.

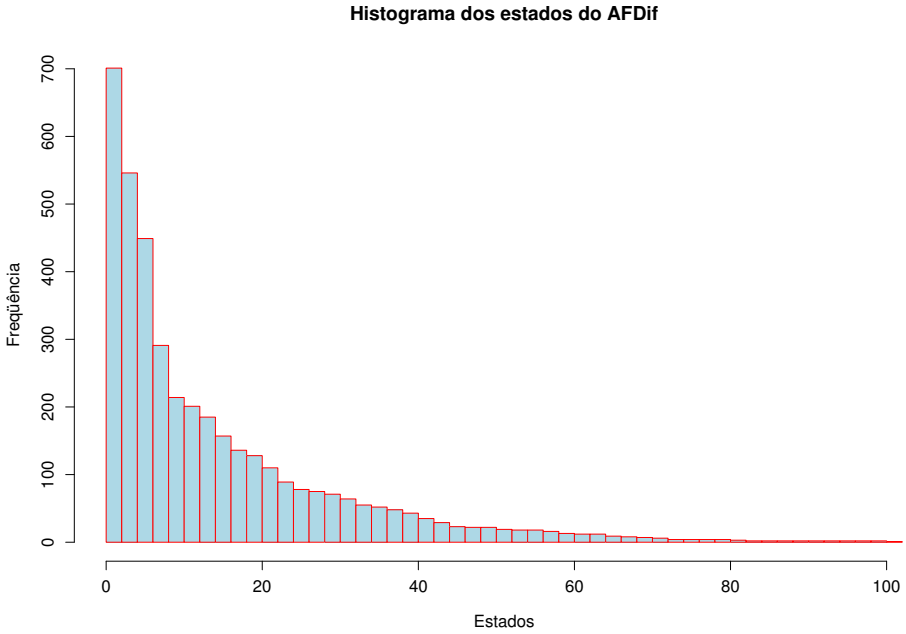

FIGURA D.4 – Histograma da densidade dos dados representados por cada um dos estados extraídos para o AFDif. O eixo x representa os estados e o eixo y, a quantidade de dados representada por cada um dos estados.

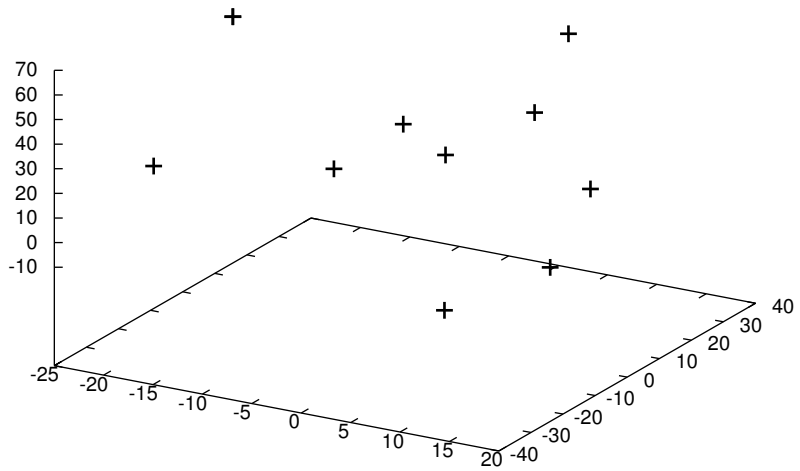

FIGURA D.5 – Centróides dos 10 estados do AFDif, plotados em um gráfico 3D, com dimenções referentes as variáveis x, y e z do sistema de Lorenz.

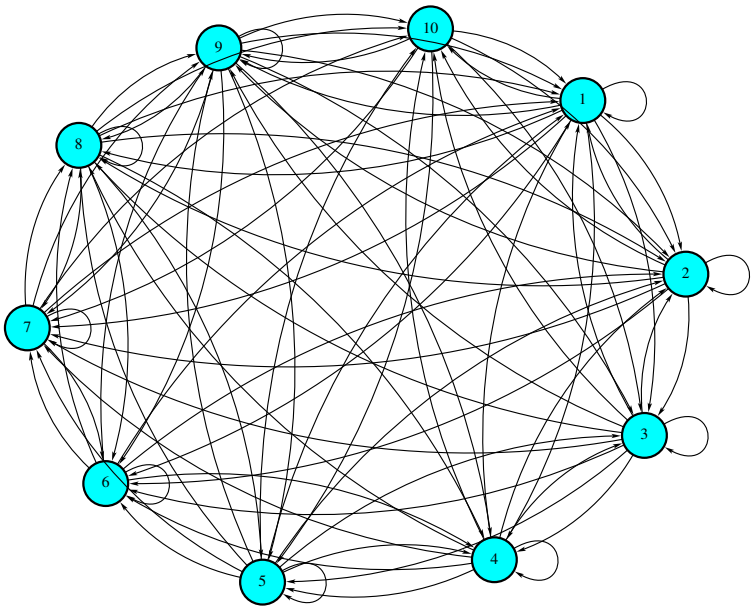

FIGURA D.6 – AFDif extraído a partir da RNR 3-11-3, com $\Delta t = 0,03$. Através da clusterização difusa foram selecionados 10 estados para a representação do sistema de Lorenz na forma de um autômato.

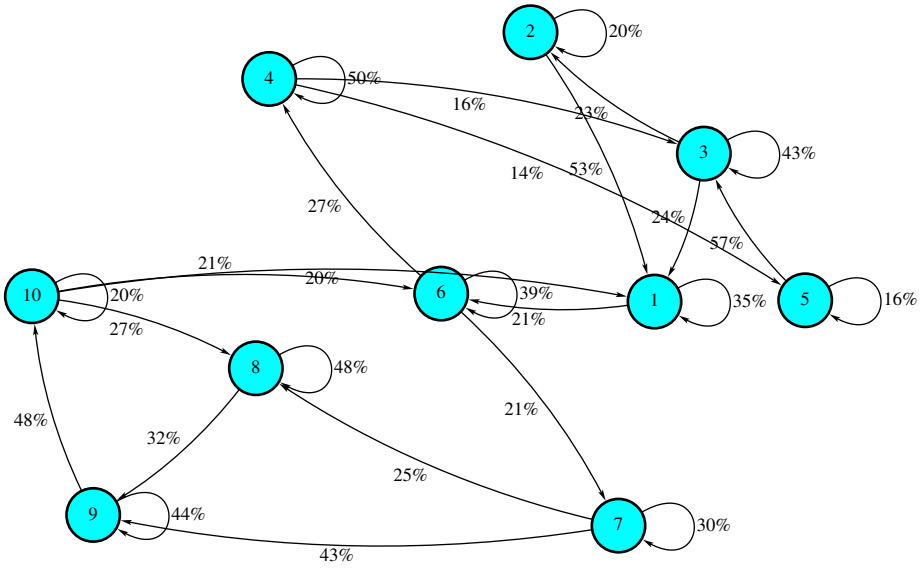

FIGURA D.7 – Cadeia de Markov extraída a partir da RNR 3-11-3, utilizando a clusterização difusa. Somente são apresentadas as transições com os valores de probabilidade mais representativos. A cadeia está amostrada considerando os valores dos centróides referentes as variáveis x e y do sistema de Lorenz.