UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

LÚCIO DORNELES MOSER

Modelo de um Neurônio Diferenciador-Integrador para Representação Temporal em Arquiteturas Conexionistas

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dr. Paulo Martins Engel
Orientador

Porto Alegre, dezembro de 2004

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Moser, Lúcio Dorneles

Modelo de um Neurônio Diferenciador-Integrador para Representação Temporal em Arquiteturas Conexionistas / Lúcio Dorneles Moser. – Porto Alegre: PPGC da UFRGS, 2004.

98 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2004. Orientador: Paulo Martins Engel.

1. Sinais temporais. 2. Derivada. 3. Integral. 4. Representação temporal. 5. Filtro digital. 6. Problemas dinâmicos. I. Engel, Paulo Martins. II. Título.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ANFIS	Adaptive-Network-based Fuzzy Inference System
ARAVQ	Adaptive Resource Allocating Vector Quantization
ARMA	Auto-Regressive Moving Average Models
BPTT	BackPropagation Through Time
LSTM	Long Short-Term Memory
MA	Moving Average
MLP	MultiLayer Perceptron
NARX	Nonlinear Auto-Regressive model with eXogenous inputs
NDI	Neurônio Diferenciador-Integrador
NRMSE	Normalized Root Mean Squared Error
RecSOM	Recursive Self-Organizing Map
RSOM	Recurrent Self-Organizing Map
RTRL	Real Time Recurrent Learning
SARDNET	Sequential Activation Retention and Decay NETwork
SOM	Self-Organizing Map
SOM-SD	Self-Organizing Map for Structured Data
SOMTAD	Self-Organizing Map with Temporal Activity Diffusion
TDNN	Time Delay Neural Network
TKM	Temporal Kohonen Map
UH	Unidade de Habituação
VLSI	Very Large Scale Integration

LISTA DE FIGURAS

LISTA DE TABELAS

AGRADECIMENTOS

A meu pai e minha mãe, Francisco e Anete, pelo amor e tranqüilidade que sempre me passaram.

A meus dois grandes irmãos, Henrique e Ramon, pela companhia e apoio que me deram.

A meu orientador, professor Engel, pela paciência, pela liberdade de exploração do tema, pelos conselhos, mas principalmente pelas longas conversas sobre IA.

Ao grupo de IA, mais especificamente Lai Yu Chin, Daniel Basso, Luís Renato Erpen, João Valiati e Edson Prestes, pelas conversas, trocas de idéias, apoio e pela camaradagem que sempre desfrutei.

Aos meus amigos que nada entendem de Inteligência Artificial, mas que torceram por mim, me incentivaram e me fizeram descansar e me divertir nos momentos certos.

Aos alunos de Lógica (2003/01) e Linguagens Formais (2003/02) por me ensinarem um pouco sobre a arte de ensinar e por me permitirem retribuir um pouco do que ganhei nesta universidade, assim como o próprio departamento de Informática Teórica que me selecionou como professor substituto.

Ao UTUG, grupo de usuários TeX da UFRGS que facilitou enormemente a criação deste documento.

Ao pessoal da biblioteca, principalmente a Ida Rossi, pela revisão das normas ABNT e a Beatriz Haro, por me ajudar nas consultas e perdoar meus atrasos na devolução dos livros!

E por fim, a todo o instituto que sempre me acolheu com muito carinho desde a graduação.

MUITO OBRIGADO!

RESUMO

O presente trabalho analisa diferentes modelos de representação temporal usados em arquiteturas conexionistas e propõe o uso de um novo modelo neural, chamado Neurônio Diferenciador-Integrador (NDI) para aplicações com processamento de sinais temporais. O NDI pode ser interpretado como um filtro digital. Seu funcionamento exige poucos recursos computacionais e pode ser de grande valia em problemas onde a solução ideal depende de uma representação baseada em mudanças e não em valores absolutos da entrada. Como vantagens desse modelo pode-se citar a representação temporal instantânea, facilidade de implementação, modularidade e eliminação de ruído. Após a definição do modelo, o mesmo é sujeito a alguns experimentos teóricos utilizado em conjunto com arquiteturas conexionistas clássicas para resolver problemas que envolvem o tempo, como previsão de séries temporais, controle dinâmico e segmentação de seqüências espaço-temporais. Como conclusão, o modelo neural apresenta grande potencialidade principalmente na robótica, onde é necessário tratar os sinais sensoriais ruidosos do robô de forma rápida e econômica.

Palavras-chave: Sinais temporais, derivada, integral, representação temporal, filtro digital, problemas dinâmicos.

A Differentiator-Integrator Neural Model for Temporal Representation in Connectionist Architectures

ABSTRACT

The present work analyzes different models for generation of temporal representation in connectionist architectures and considers the use of a new neural model, called Differentiator-Integrator Neuron (NDI) for applications in processing of temporal signals. The NDI can be interpreted as a digital filter. Its functioning demands few computational resources and can be of great value in problems where the ideal solution depends on a representation based on changes and not on absolute values of the input. The instantaneous temporal representation, easiness of implementation, modularity and elimination of noise are the principal advantages of this model. After defining the model, we tested it with classic connectionist architectures in some theoretical experiments to solve problems that involve the time, such as forecast of temporal series, dynamic control and segmentation of spatial-temporal sequences. The results indicate that the neural model presents great potentiality mainly in the robotics, where it is necessary to process the noisy sensorial signals of the robot in a quick and economic way.

Keywords: Temporal Signals, Derivative, Integral, Temporal Representation, Digital Filter, Dynamic Problems.

1 INTRODUÇÃO

A representação de dados é o cerne dos problemas das arquiteturas conexionistas. Qualquer rede neural pode ser interpretada como uma função de mapeamento não linear entre os dados de entrada e os dados de saída (HAYKIN, 1999). A representação dos dados de entrada e dos dados de saída tem papel fundamental sobre a solução encontrada podendo, muitas vezes, ser a causa do insucesso no aprendizado. Em geral, a representação dos dados não é discutida e planejada com o devido cuidado ao se projetar uma rede neural. A atenção se volta aos parâmetros da rede, como o número de neurônios nas camadas ocultas, a taxa de aprendizado, etc. Talvez este descaso seja conseqüência da imagem geral que existe sobre as redes neurais. As redes neurais são consideradas caixas pretas que resolvem os problemas sem a necessidade de nenhum conhecimento sobre seus domínios.

Os algoritmos de aprendizado para redes neurais buscam relações estatísticas entre os dados de entrada e os de saída. Quando os problemas são complexos, e isto ocorre freqüentemente, não existem relações estatísticas significativas entre os dados de saída e os de entrada. A relação é indireta. Nestes problemas são necessárias redes com camadas ocultas, para que os dados de entrada sejam recodificados em espaços de representação alternativos e que então apresentem relação estatística (mapeamento linear) com os dados de saída. Estes problemas são os chamados, não linearmente separáveis. A descoberta da representação intermediária é feita utilizando-se algoritmos que seguem o gradiente descendente do erro, como a retropropagação de erro. O espaço de soluções é infinito e totalmente dependente das representações originais dos dados de entrada e de saída. A busca deve ser gradual e lenta para que a superfície do erro varie pouco entre uma solução e outra. Este tipo de abordagem pode levar a mínimos locais que só podem ser evitados com mais de uma simulação, variação dinâmica da taxa de aprendizado, ou qualquer outra heurística de aprendizado (CLARK; THORNTON, 1997).

Elman realizou uma série de experimentos em problemas de aquisição de gramática aplicando redes neurais recorrentes simples (ELMAN, 1993). Suas conclusões foram de que o aprendizado só é bem sucedido com o uso de heurísticas, ou por meio de tratamento adequado dos dados de treinamento como a apresentação de casos simples antes dos casos complexos, ou por meio de ampliação gradual dos recursos da rede, como aumento da camada de neurônios de entrada. Ambos os métodos fazem com que a rede descubra representações mais genéricas em primeiro lugar, e a medida que elas estabilizam, a rede se especializa nos casos complexos, criando representações específicas. Estes métodos são heurísticas de aprendizado que auxiliam a busca por representações internas e são muito eficazes para dados que possuem estrutura hierárquica implícita.

É evidente que as redes neurais não resolvem qualquer problema, independente da representação escolhida. Deve-se utilizar todo o conhecimento disponível para facilitar seu trabalho na busca por representações internas, ou utilizando heurísticas de aprendizado ou escolhendo adequadamente a representação dos dados de entrada. Este trabalho está voltado para a escolha adequada de representação dos dados de entrada em problemas dinâmicos, ou seja, onde o tempo também deve ser representado pois é parte do mapeamento desejado. É proposto uma representação temporal alternativa com o objetivo de facilitar o aprendizado e melhorar a generalização em problemas que envolvem o processamento de sinais com correlação temporal e ruído.

1.1 Motivação

O presente trabalho tem como foco aplicações de robótica. Os problemas encontrados na robótica são, em geral, bastante complexos, pois envolvem grande dimensionalidade de dados e demandam processamento multisensorial em tempo real. Os dados são espaço-temporais e apresentam muita redundância e ruído. Além disso, a interação com o ambiente pode levar a situações nunca vistas, e portanto, é necessário constante aprendizado e adaptação. Geralmente um robô deve atender mais de um objetivo (por exemplo, auto-localização, evitar colisões, encontrar objetos, etc.) e não existe uma única solução, mas um conjunto infinito de soluções possíveis. Nestes casos, o aprendizado mais adequado é o chamado aprendizado por reforço, onde o robô recebe reforços positivos ou negativos somente quando cumpre os objetivos predeterminados ou quando realiza ações inapropriadas. O robô deve associar corretamente o reforço recebido às suas ações anteriores e aos estados que passou, para que, em outras circunstâncias, aja de forma mais adequada (BARTO; SUTTON; ANDERSON, 1983; SUTTON, 1988). Se os sinais de reforço forem muito raros, será difícil encontrar suas causas verdadeiras, pois haverá muitas ações e estados que os precederam. Este problema é conhecido como problema de atribuição de créditos e sua maior conseqüência é o impacto no tempo de convergência para a solução ótima (HAYKIN, 1999).

Apesar de inerentemente simbólicos, os algoritmos de aprendizado por reforço podem ser adaptados para uso em redes neurais recorrentes (ANDERSON, 1986). As redes neurais, por terem recursos limitados, representam somente os estados mais significativos para a resolução da tarefa, encontrando descrições compactas do estado do ambiente. Mesmo assim, a dificuldade para encontrar soluções é muito grande, pois as redes neurais recorrentes utilizam treinamentos derivados da retropropagação de erro que, por buscarem relações temporais além das espaciais, apresentam complexidade algorítmica muito maior que a complexidade do algoritmo de retropropagação de erro padrão. Tais algoritmos, como o BPTT (do inglês,*Back Propagation Through Time*), o RTRL (do inglês, *Real Time Recurrent Learning*) e derivados são computacionalmente complexos e já foi comprovado que não relacionam dados mais distantes que dez instantes de tempo, ou seja, se alguma informação for necessária ser mantida em memória por mais de dez instantes para solucionar um determinado problema (problemas de dependência de longo prazo), este problema não será resolvido pelos métodos tradicionais (HOCHREITER; SCHMIDHUBER, 1997).

A ineficiência das arquiteturas conexionistas que se propõem a resolver problemas temporais como este pode ser compreendida pelas dificuldades inerentes do processamento temporal. Talvez a maior delas seja a falta de limites explícitos entre

os padrões temporais. Em problemas estáticos, os padrões espaciais têm tamanho limitado, por exemplo, imagens de caracteres para reconhecimento de escrita. A rede neural deve obter uma representação interna a partir de cada imagem apresentada e classificá-la como um dos símbolos válidos. Em problemas temporais, o tamanho dos padrões e os instantes em que começam e terminam devem ser descobertos. Por exemplo, em uma aplicação de processamento de voz, a menos que haja algum pré-processamento do sinal sonoro, não há limites explícitos entre os fonemas, palavras ou sentenças. No entanto todos estes conceitos deveriam ser identificados para a obtenção de uma representação compacta e eficiente da fala. Além disso, a rede deve memorizar o sinal sonoro enquanto o padrão temporal está sendo processado. Somente após a apresentação integral do padrão temporal é possível fazer sua classificação. As redes neurais devem então, aprender um espaço de representação que memorize a entrada de dados por tempo suficiente e sem ambigüidades para que ao final seja possível distinguir qual padrão temporal ocorreu na entrada. Esta tarefa é extremamente custosa e envolve a busca por relacionamentos no domínio tempo. Como já foi dito, esse relacionamento não passa de dez unidades temporais em arquiteturas tradicionais, o que limita consideravelmente o conjunto de problemas que podem ser resolvidos por estes métodos.

As limitações impostas pelas redes neurais recorrentes podem ser contornadas se os dados de entrada forem acompanhados de informação temporal útil. Desta forma, a tarefa de encontrar representações internas se torna mais simples e os problemas dinâmicos tratáveis podem ser mais complexos, como os existentes na robótica. As técnicas de representação temporal abordadas na revisão bibliográfica deste trabalho podem ser utilizadas, não só na robótica, mas em muitos outros problemas dinâmicos, como problemas de reconhecimento de fala, identificação de sons, detecção de movimentos, rastreamento de alvos, reconhecimento de objetos em tempo real, movimentos sincronizados, processos de controle dinâmico e previsão de séries temporais.

A motivação deste trabalho é a busca por representações temporais úteis ao processamento de sinais temporais. Entende-se por útil, uma representação que permita a generalização de conceitos e a rápida adaptação a mudanças na dinâmica do sistema, mas sobretudo, que seja econômica sob o ponto de vista computacional. Essa representação deve ser computada em tempo real e se aproveitar das características dos sinais temporais, como a correlação temporal.

1.2 Objetivos

Os objetivos deste trabalho são:

- Resumir tecnologias empregadas na Inteligência Computacional para a criação de representação temporal que possam ser úteis nos problemas da robótica ou problemas dinâmicos complexos.
- Sugerir um modelo neural simples mas bastante eficiente que possa ser utilizado em qualquer arquitetura conexionista para extrair informações dinâmicas dos dados de entrada e possivelmente melhorar o espaço de representação do fluxo de dados de entrada.
- Aplicar o modelo neural sugerido em conjunto com arquiteturas conexionistas a alguns problemas temporais conhecidos e comparar com soluções típicas da

Inteligência Computacional para determinar sua utilidade e potencialidades.

O modelo de neurônio proposto neste trabalho foi planejado para atender os seguintes requisitos:

- seja uma solução biologicamente plausível, mas sobretudo econômica sob o ponto de vista computacional;
- seja modular, para que possa ser aplicado em cadeia, gerando informações de diferentes níveis de abstração.
- permita a utilização em qualquer arquitetura conexionista (perceptron de múltiplas camadas, mapas auto-organizáveis, redes recorrentes, etc.).
- tenha parâmetros para se adaptar às características do sinal.

Espera-se com este trabalho, esclarecer e ampliar as formas de representação temporal existentes aplicáveis aos modelos conexionistas.

1.3 Organização

O Capítulo 2 é uma revisão abrangente dos modelos de representação temporal voltados ao processamento de sinais temporais uni ou multidimensionais para aplicação em arquiteturas conexionistas.

O Capítulo 3 apresenta o modelo computacional do neurônio (Neurônio Integrador-Diferenciador), bem como análises teóricas do modelo e argumentação sobre sua plausibilidade biológica.

O Capítulo 4 relata os resultados de diferentes testes aplicados em arquiteturas conexionistas associadas com o Neurônio Integrador-Diferenciador e comparações com resultados de outras arquiteturas.

O Capítulo 5 apresenta conclusões e sugere trabalhos futuros baseados no novo modelo neural.

2 MODELOS DE REPRESENTAÇÃO TEMPORAL

Em todos os problemas dinâmicos, a computação da saída deve levar em conta eventos passados, e para isso, o sistema de aprendizado deve possuir memória. Como já foi visto na introdução, as redes neurais recorrentes possuem poder limitado para a memorização de eventos. Ao invés disso, poder-se-ia utilizar componentes de memória na entrada da rede. Estes componentes tornam explícitas as informações temporais dos dados de entrada, facilitando a tarefa da rede neural em descobrir relações espaço-temporais.

Este capítulo resume modelos de memória e modelos conexionistas que geram representação temporal a partir de seqüências de dados uni ou multidimensionais. Esta representação pode ser adicionada aos dados de entrada de arquiteturas conexionistas para resolver os problemas dinâmicos de forma mais simples. A figura 2.1 demonstra como as representações temporais podem ser aplicadas na robótica. O ambiente é regido por um sistema dinâmico que possui um estado E(t) e regras próprias de transição de estado. O estado do ambiente pode ser afetado pelas ações do robô. O robô percebe o ambiente através de seus sensores, coletando observações parciais do estado do sistema (O(t)). Estas observações são em geral dados estáticos obtidos em instantes de tempo com intervalos fixos, como imagens de vídeo, distância de objetos ao redor, ângulo das articulações mecânicas, posição no ambiente, etc. Estas observações não são suficientes para se reconstituir o estado completo do sistema. A representação temporal gerada pelo acúmulo de observações e ações do robô juntamente com a última observação do ambiente é aplicada a entrada da arquitetura conexionista para que esta gere uma representação interna análoga aos estados reais do sistema (E*(t)). Uma vez representado o estado, o sistema pode facilmente mapear para as ações corretas.

O problema de auto-localização em labirintos é um exemplo típico, no âmbito da robótica, de problema dinâmico. Supõe-se um labirinto constituído por corredores de mesma espessura e um robô que conhece o mapa topológico do ambiente mas não sua localização inicial. Para que o robô descubra sua posição, não basta analisar os sinais sensoriais, pois os corredores são exatamente iguais. Deve na verdade, transitar pelo ambiente, coletando sua estrutura espacial no tempo (corredor, curva a direita, bifurcação, etc.) até que a seqüência obtida identifique sem ambigüidades a posição que ocupa no labirinto. Neste momento, terá descoberto o estado real do sistema e poderá tomar a menor rota para a saída do labirinto.

Existem problemas temporais na robótica que possuem natureza simbólica e exigem a memorização explícita de eventos remotos. São problemas de dependência de

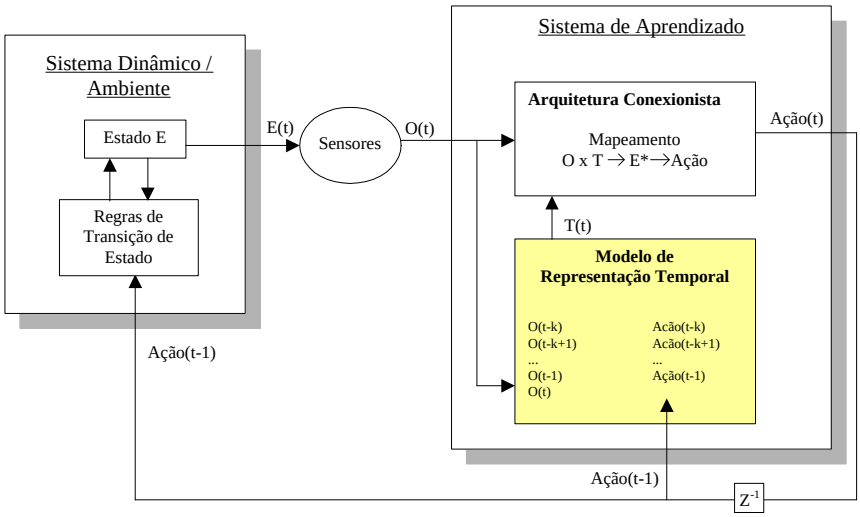

Figura 2.1: Uso de representação temporal aplicada a sistema de aprendizado em ambientes dinâmicos.

longo prazo. Tarefas como limpeza de ambientes, ou recuperação de objetos perdidos exigem que o robô saiba exatamente todos os locais por onde já passou para não repetir seu trabalho. Este tipo de problema tem natureza simbólica pois pode ser formalizado como um problema de reconhecimento de gramáticas. Tais problemas dificilmente se favoreceriam pelo uso das técnicas de representação temporal abordadas neste capítulo, pois a memória deve ser utilizada para reter informações específicas do passado e não simplesmente descrever os últimos eventos percebidos. Recentemente foi sugerido o modelo conexionista chamado LSTM (do inglês, *Long Short-Term Memory*) que é bastante adequado a problemas simbólicos. O LSTM utiliza componentes de memória programáveis que podem reter a informação por longos períodos de tempo. O LSTM resolveu problemas de reconhecimento de gramáticas livre do contexto que nunca foram solucionados por outras arquiteturas conexionistas (GERS et al., 2002) e portanto tem grande potencialidade em problemas de natureza simbólica. No entanto, em domínios contínuos como os da robótica, sua aplicação é praticamente inviável (GERS; ECK; SCHMIDHUBER, 2001), a menos que utilizado em conjunto com técnicas de redução de dimensionalidade (BAKKER; LINåKER; SCHMIDHUBER, 2002).

Os problemas aos quais o presente trabalho está voltado são aqueles onde há o tratamento direto do fluxo sensório-motor de robôs. Os modelos de representação temporal apresentados são analisados levando-se em conta dados de entrada contínuos de grande dimensionalidade, com redundância, correlação espaço-temporal e perturbações de ruído. Apesar de bastante diferentes, todas as soluções abordadas possuem o mesmo objetivo: representar a informação temporal implícita nos dados de entrada, de forma explícita, ocupando posições no espaço. Espera-se com isso, que a partir de um problema dinâmico qualquer, os dados estáticos (observações do ambiente), juntamente com a representação temporal gerada, sejam suficientes para reconstituir o estado do sistema dinâmico, e finalmente, permitir o mapeamento direto entre estado e ação apropriada.

As seções deste capítulo apresentam diferentes modelos de representação temporal divididos pelo método de criação da representação, que pode ser: predefinido, como nas memórias de curto prazo, gerado com conceitos de auto-organização, como

nos mapas auto-organizáveis ou gerado a partir do princípio de compressão de história, como em arquiteturas conexionistas hierárquicas.

Como não existem métricas rígidas para se avaliar a qualidade de uma representação temporal, a maioria dos modelos apresentados foi implementada em MATLAB e testada em casos simples. O objetivo da análise foi o de obter uma idéia geral das características de cada modelo de representação, levando-se em conta a capacidade de distinção de seqüências temporais (análise de ambigüidade), a dificuldade na determinação de parâmetros adequados, a capacidade de convergência e estabilidade da representação gerada, o custo computacional e a sensibilidade a perturbações tipicamente encontradas em dados sensoriais.

2.1 Memórias de Curto Prazo

As memórias de curto prazo são unidades que processam seqüências temporais unidimensionais. São responsáveis pela descrição de aspectos importantes da seqüência de entrada referentes aos últimos instantes de tempo. As memórias não apresentam aprendizado. Seu comportamento é predeterminado por seus parâmetros de funcionamento. Existem diversos tipos de memória, mas as principais serão examinadas nesta seção bem como um recente modelo proposto, chamado de Unidade de Habituação. A análise dos primeiros três tipos de memória é baseada no trabalho de Mozer (MOZER, 1993), um bom estudo comparativo de arquiteturas conexionistas temporais, onde é sugerida uma taxonomia básica para a comparação dos modelos existentes.

Após apresentados os modelos de memória, é feita uma comparação entre os tipos de representação temporal gerados por cada um, visando problemas de ambigüidade e de sensibilidade a deslocamentos temporais.

2.1.1 Memória de Atraso

A forma mais simples de representar informação temporal é memorizar os últimos valores dos dados de entrada e apresentá-los como se fossem informações independentes. As Memórias de Atraso seguem este princípio. Cada memória mantém o valor de um instante passado do sinal temporal. Sua representação gráfica é dada pela figura 2.2. O parâmetro k determina o atraso da memória, ou seja, de quanto tempo o sinal da saída está defasado em relação ao sinal de entrada.

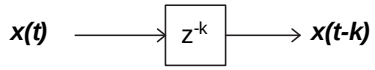

Figura 2.2: Representação gráfica de uma memória de atraso k.

Para representar mais de uma informação passada, as memórias de atraso devem ser encadeadas, formando linhas de atraso. A figura 2.3 ilustra uma seqüência temporal e sua representação final na linha de atraso com 4 memórias. A linha de atraso representa de forma explícita e sem perda de informação os últimos 5 instantes de tempo da seqüência temporal da entrada. No entanto, todos os instantes anteriores não são representados. A informação temporal gerada pelas linhas de atraso sofre um truncamento abrupto.

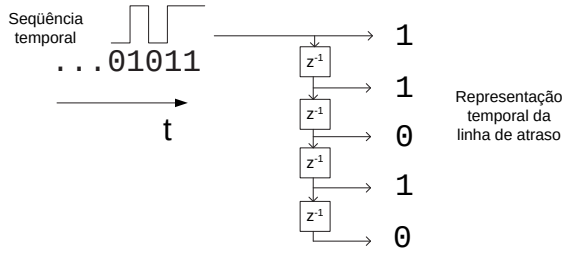

Figura 2.3: Representação de sinal temporal em uma linha de atraso de 4 memórias.

Este tipo de memória é provavelmente o mais utilizado em arquiteturas conexionistas. O uso de linhas de atraso em dados de entrada de uma rede neural faz com que um problema dinâmico se torne um problema estático. Os padrões temporais são representados como padrões espaciais. Desta forma, a rede neural utilizada pode ser um *Perceptron de Múltiplas Camadas* (MLP) tradicional com treinamento por retropropagação de erro, que não leva em conta relacionamentos temporais. A complexidade, tanto do processamento das linhas de atraso, como do treinamento da rede é relativamente pequena. Exemplos de arquiteturas conexionistas deste tipo, são o TDNN (do inglês, *Time Delay Neural Network*) e o NARX (do inglês, *Nonlinear Auto-Regressive model with eXogenous inputs*) (HAYKIN, 1999).

Para compreender o uso das linhas de atraso, pode-se tomar o problema do XOR temporal como exemplo. Este problema foi sugerido por Elman (ELMAN, 1990) para estudar o comportamento de sua rede recorrente. O problema consiste em uma versão dinâmica do problema XOR. O problema XOR é conhecido por ser simples mas não linearmente separável. A rede deve aprender o mapeamento dos padrões espaciais binários 0,0 e 1,1 para a saída 0 e os padrões 1,0 e 0,1 para a saída 1. Na versão temporal, os dois bits de entrada são apresentados seqüencialmente e o resultado da operação XOR sobre eles é o terceiro bit, como na seqüência abaixo.

1 0 1 0 0 0 0 1 1 1 1 0 1 0 1 ...

A rede é treinada para prever o próximo bit da seqüência de entrada, mas como os dois primeiros bits de cada conjunto de três são gerados aleatoriamente, a rede somente terá sucesso de previsão no terceiro bit de cada conjunto, correspondente ao resultado das operações XOR. A rede recorrente de Elman resolveu o problema gerando representações internas para as quatro possíveis combinações dos dois últimos bits de entrada: 0,0; 0,1; 1,0 e 1,1, reduzindo a um problema estático. Esta mesma abordagem pode ser feita com linha de atraso, com a vantagem de que a representação é instantânea. Neste caso, a rede neural utilizada pode ser um MLP comum. A figura 2.4 ilustra a arquitetura e como a seqüência temporal adquire representação estática, reduzindo o problema temporal à versão original do problema XOR.

Deve-se ter cuidado, portanto, na escolha do tamanho da linha de atraso utilizada. Se for menor que os padrões temporais existentes nos dados, a rede neural não terá como identificá-los sem ambigüidades. Se for muito maior, a rede neural terá dificuldades em criar generalizações, ou seja, deverá aprender as mesmas associações para diversos casos semelhantes, utilizando recursos independentes e um número elevado de épocas de treinamento.

Para relacionar o comportamento deste tipo de memória com os demais, será introduzido um modelo matemático que generaliza a computação realizada pelas

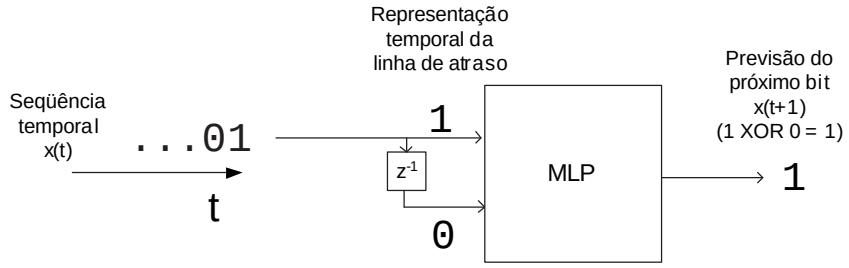

Figura 2.4: Uso de linha de atraso no problema do XOR temporal.

memórias tal qual foi utilizado por Mozer (MOZER, 1993). O sinal de saída das memórias pode ser expresso em função do sinal de entrada. Esta função é apresentada na equação 2.1 e representa a convolução dos valores de entrada $x(t)$ em cada instante t por pesos de participação $c(t)$. Os pesos de participação $c(t)$, também conhecidos como função *kernel*, caracterizam o comportamento de cada tipo de memória.

$$\overline{x}(t) = \sum_{\tau=1}^{t} c(t-\tau)x(\tau) \tag{2.1}$$

A memória de atraso mantém o conteúdo exato do último sinal de entrada, por isso a função $c(1)$ é 1 e para todos os demais valores de t é 0, como ilustrado em 2.5. É importante notar, que o parâmetro t para a função *kernel* se refere a tempo decorrido. As arquiteturas conexionistas geralmente empregam memórias com atraso unitário, mas no capítulo 4 são utilizadas memórias com atraso de 6 instantes de tempo. Neste caso, a função *kernel* $c(t)$ é 1 somente para $t = 6$.

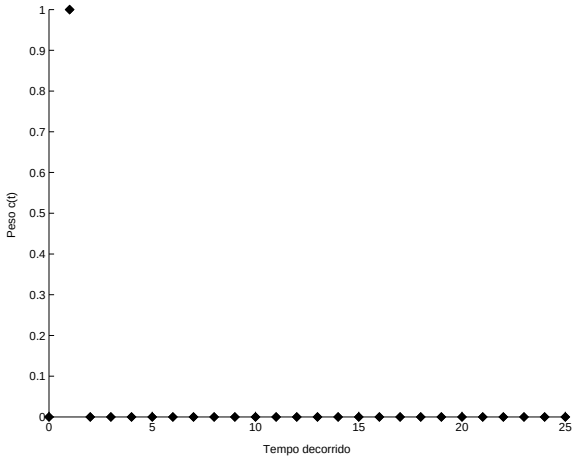

Figura 2.5: Função *kernel* da memória de atraso unitário.

2.1.2 Memória de Traço Exponencial

Diferente das memórias de atraso, as memórias de traço exponencial não perdem informação abruptamente a um determinado ponto no tempo. Ao invés disso, a informação degrada suavemente. As entradas mais recentes têm maior representação que as mais distantes no tempo. A vantagem destas memórias é a facilidade de computação incremental. A cada instante de tempo, seu conteúdo pode ser determinado

pelo sinal de entrada atual e a sua atividade no instante anterior. A atualização da memória é dada pela equação 2.2. Onde $x(t)$ é o sinal de entrada no tempo t e μ é a constante de decaimento entre 0 e 1. Considera-se a condição inicial $\overline{x}(0) = 0$. A figura 2.6 representa graficamente a memória exponencial.

$$\overline{x}(t) = (1 - \mu)x(t) + \mu\overline{x}(t - 1) \tag{2.2}$$

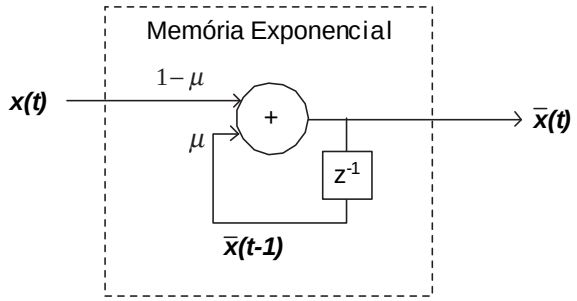

Figura 2.6: Modelo da Memória Exponencial.

A memória exponencial pode ser analisada com base em sua função *kernel* $c(t)$, definida pela equação 2.3.

$$c(t) = (1 - \mu)\mu^t \tag{2.3}$$

Este tipo de memória é um caso especial dos modelos estatísticos tradicionais de média móvel ou MA (do inglês, *Moving Average Models*) (MOZER, 1993). Seu comportamento pode ser compreendido pela figura 2.7, que representa a função *kernel* $c(t)$ para alguns valores de μ. Pela figura é fácil perceber como a informação das entradas anteriores decai conforme a passagem do tempo gerando uma espécie de média móvel. O parâmetro μ determina a sensibilidade aos eventos passados. Em última análise, a memória exponencial realiza integração temporal sobre os dados de entrada, ou seja, seu valor de saída representa um valor acumulado do sinal de entrada.

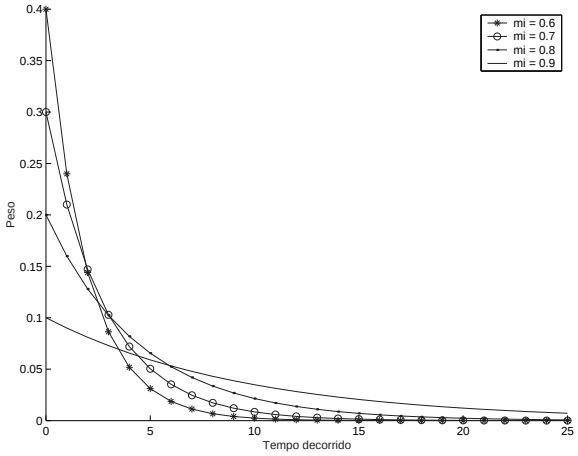

Figura 2.7: Função *kernel* das memórias exponenciais para $\mu = 0{,}6; 0{,}7; 0{,}8$ e $0{,}9$.

Tabela 2.1: Valores na saída da memória exponencial de$\mu = 0{,}2$ para cada seqüência de dois bits no problema do XOR temporal

$x(t-1)$	$x(t)$	$\overline{x}(t)$
0	0	de 0 até 0,04
1	0	de 0,16 até 0,2
0	1	de 0,8 até 0,84
1	1	de 0,96 até 1

A equação 2.4 corresponde à mesma função da equação 2.2. No entanto, esta forma evidencia o comportamento dinâmico de$\overline{x}$. O valor da memória se aproxima do valor de entrada atual $x(t)$ a uma taxa constante $(1-\mu)$.

$$\overline{x}(t) = \overline{x}(t-1) + (1-\mu)(x(t) - \overline{x}(t-1)) \tag{2.4}$$

Teoricamente a memória poderia representar qualquer seqüência temporal binária sem perda de informação, pois sua função *kernel* é diferente de zero para qualquer t. No entanto, a resolução finita e a existência de ruído fazem com que os bits menos significativos da representação temporal, correspondentes aos eventos mais distantes, sejam perdidos (MOZER, 1993).

O problema do XOR temporal já discutido também poderia ser solucionado pelo uso de memórias exponenciais. Apesar da integração realizada pela memória envolver todos os bits da seqüência, os últimos bits são mais significativos e podem ser recuperados sem ambigüidades. A equação da memória pode ser desdobrada para depender dos dois últimos valores de entrada $x(t)$ e $x(t-1)$, resultando na equação 2.5. Se for utilizado uma memória exponencial com constante de decaimento$\mu = 0{,}2$ resulta na equação 2.6

$$\overline{x}(t) = (1-\mu)x(t) + \mu[(1-\mu)x(t-1) + \mu\overline{x}(t-2)] \tag{2.5}$$

$$\overline{x}(t) = 0{,}8x(t) + 0{,}16x(t-1) + 0{,}04\overline{x}(t-2) \tag{2.6}$$

A tabela 2.1 apresenta o intervalo dos valores na saída da memória para cada seqüência de dois bits, gerada a partir da equação 2.6. O resultado é um intervalo pois o valor de saída depende do valor da própria memória anterior aos dois bits $(\overline{x}(t-2))$, e este se encontra no intervalo de 0 a 1, o mesmo intervalo dos dados de entrada.

Pela tabela pode-se concluir que a memória representa os últimos dois bits da seqüência sem ambigüidades e portanto, é possível resolver o problema com um MLP. Basta que este divida o espaço de entrada (produzido pela memória exponencial) nas faixas de valores correspondentes às combinações dos dois últimos bits e mapeie para o valor de saída correto.

2.1.3 Memória Gama

A memória Gama é um modelo parametrizável que generaliza os dois modelos anteriores. Na representação gráfica da figura 2.8 claramente pode-se identificar unidades de memória exponencial encadeadas. As memórias exponenciais são governadas pelo parâmetro μ, e a quantidade de memórias empregadas é determinado

pelo parâmetro ω. Estes dois parâmetros definem um contínuo de variações entre as memórias de atraso e as memórias exponenciais.

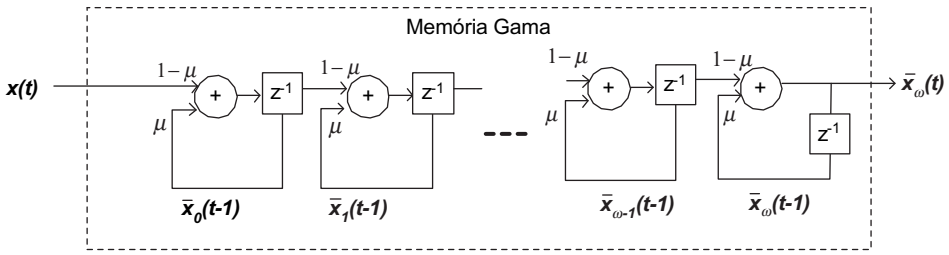

Figura 2.8: Representação gráfica da memória Gama.

A equação 2.7 define o cálculo de atualização da memória Gama. Para calcular o valor de saída de uma memória $\overline{x}_\omega(t)$ é necessário calcular também $\overline{x}_j(t)$ para j = 0,1,2,3...,$\omega - 1$. Considera-se as seguintes condições iniciais: $\overline{x}_{-1}(t) = x(t+1)$ para todo $t \geq 0$, e $\overline{x}_j(0) = 0$ para todo $j \geq 0$.

$$\overline{x}_j(t) = (1-\mu)\overline{x}_{j-1}(t-1) + \mu\overline{x}_j(t-1) \tag{2.7}$$

A função *kernel* da memória Gama é definida pela equação 2.8 e é representada na figura 2.9 para os parâmetros $\omega = 6$ e $\mu = 0{,}3; 0{,}5$ e $0{,}7$. Pela figura pode-se perceber como a memória agrega a funcionalidade das duas memórias básicas.

$$c(t) = \begin{cases} \binom{t}{\omega}(1-\mu)^{\omega+1}\mu^{t-\omega} & \text{se } t \geq \omega \\ 0 & \text{se } t < \omega \end{cases} \tag{2.8}$$

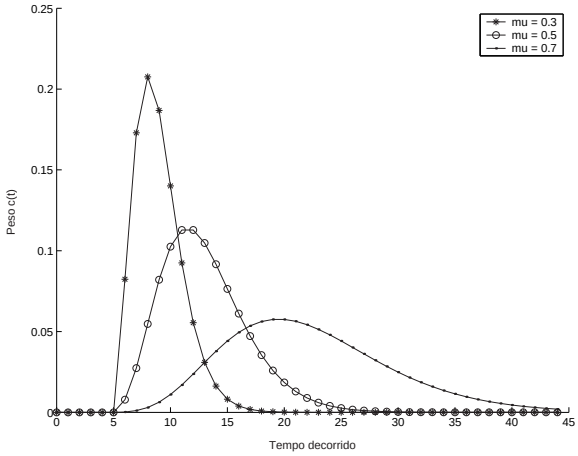

Figura 2.9: Função *kernel* das memórias Gama para $\omega = 6$ e $\mu = 0{,}3; 0{,}5$ e $0{,}7$.

2.1.4 Unidades de Habituação

A habituação é um fenômeno tipicamente encontrado em sistemas neurais biológicos. É o processo pelo qual pesos sinápticos de neurônios são alterados para ignorar

estímulos repetitivos e irrelevantes, funcionando como um filtro de novidade. Este fenômeno já foi usado como fonte de inspiração em inúmeros modelos conexionistas como uma forma de representação temporal, sempre associado aos pesos sinápticos (ARBIB, 1995).

Em (STILES; GHOSH, 1995), é sugerido o uso de unidades de habituação. Estas unidades são componentes básicos, como as memórias exponenciais, que transformam o espaço de representação de entrada levando em conta as atividades anteriores. A transformação é definida pela equação 2.9.

$$y(t) = y(t-1) + \tau(\alpha(1 - y(t-1)) - y(t-1)x(t)) \tag{2.9}$$

Onde $x(t)$ é o sinal de entrada, τ determina a velocidade de habituação e α determina a razão entre velocidade da habituação e a velocidade da recuperação da habituação. Considera-se a condição inicial $y(0) = 1$. A representação gráfica desta equação é ilustrada na figura 2.10.

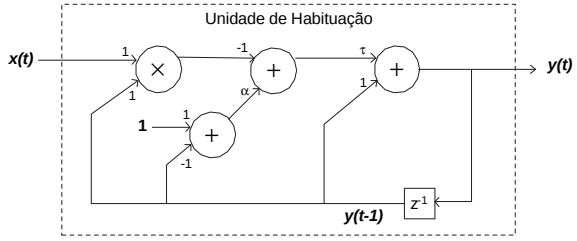

Figura 2.10: Representação gráfica da unidade de Habituação.

As unidades de habituação não podem ser expressas com a convolução por alguma função *kernel*, pois o sinal de entrada não é multiplicado por pesos constantes, mas pelo valor da própria memória (fator $y(t-1)x(t)$ da equação 2.9). Este tipo de memória faz parte de uma família de modelos mais genérica que as médias móveis, denominada ARMA (do inglês, *Auto-Regressive Moving Average Models*) (BOX; JENKINS; REINSEL, 1994).

As unidades de habituação são diferentes das memórias exponenciais. Nas memórias exponenciais, o valor de saída tende a uma taxa constante em direção a última entrada de dados. Na unidade de habituação, existem dois fatores conflitantes. O fator $(1 - y(t-1))$ leva a unidade ao valor 1, correspondendo ao estado de não habituação. E o fator $-y(t-1)x(t)$, por outro lado, leva a unidade ao valor 0, que corresponde ao estado de habituação. Qualquer sinal de entrada diferente de0 faz com que a memória tenda a 0. Esse viés de representação pode ser importante em algumas situações como apresentado na próxima subseção.

Para fins de representação temporal, pouco importa o significado original do fenômeno de habituação. O importante é o comportamento dinâmico da memória frente à seqüência temporal de entrada, ou seja, a representação gerada para cada padrão temporal e a sensibilidade a variações do sinal.

2.1.5 Ambigüidade e variabilidade de representação das memórias de curto prazo

O tipo de representação temporal gerado pelas memórias é uma questão pouco discutida e pesquisada. É importante conhecer suas limitações e potencialidades para decidir quando e como aplicá-las em problemas dinâmicos. Como já foi dito, as

memórias de curto prazo representam os últimos instantes da seqüência temporal observada. Mas cada tipo de representação reflete propriedades temporais específicas, ocasionando perda de informação, o que necessariamente gera ambigüidades.

A questão da ambigüidade é fundamental, pois em determinados casos ela é algo desejável. Por exemplo, se o sinal possuir ruído, deseja-se obter a mesma representação temporal independente deste. Da mesma forma, a ambigüidade é desejável para outras perturbações encontradas em sinais temporais, como a variação da velocidade do padrão temporal, a mudança de amplitude ou deslocamentos temporais. A escolha pela melhor representação depende do problema e do tipo de sinal temporal disponível, bem como das perturbações existentes no sinal temporal.

A linha de atraso representa sem ambigüidades os últimos k sinais de entrada, sendo $k-1$ o tamanho da linha de atraso e considerando memórias de atraso unitário. Se as seqüências temporais forem maiores que k, necessariamente existirá ambigüidade na representação de todas aquelas que tiverem os últimos k valores iguais. Por exemplo, as seqüências temporais binárias 0,0,0,1,0,1 e 1,1,0,1,0,1 terão mesma representação em linhas de atraso de até 3 memórias. A linha de atraso deve ser projetada tendo em vista o tamanho mínimo dos padrões temporais mais simples para que não haja ambigüidade de representação.

O segundo problema das linhas de atraso foi apontado por Elman em (ELMAN, 1990) e diz respeito a sensibilidade quanto à posição temporal dos padrões. Pode-se dizer que as seqüências temporais 0,1,0,1,0,1,0,1 e 1,0,1,0,1,0,1,0 representam o mesmo padrão deslocado em um instante de tempo. Apesar de se tratar do mesmo padrão temporal, as representações geradas pela linha de atraso são ortogonais. Para que uma arquitetura conexionista considere estas duas representações um mesmo padrão temporal, seria necessário duas camadas neurais. Na primeira, há o reconhecimento independente das duas representações, e na segunda há a soma dos dois através de uma operação OR, para a unificação do conceito. Fica claro que a representação temporal das linhas de atraso não facilita a solução de problemas em que seja necessário a invariância quanto a posição temporal.

As memórias exponenciais compactam a informação temporal do sinal numa única representação em forma de média móvel. O deslocamento temporal de um padrão não gera representações muito distintas, pois a média sofre pouca perturbação com pequenos deslocamentos temporais. Em compensação, a representação gerada pela memória exponencial é determinada predominantemente pelos sinais mais recentes. Quanto mais distante no tempo for a informação, menos ela estará representada na saída da memória. Portanto, as memórias exponenciais também possuem um tamanho máximo de padrão temporal que pode ser distinguido sem ambigüidades. A determinação deste tamanho não é explícita como nas linhas de atraso, mas implícita através da escolha da taxa de decaimento μ. Ainda assim, mesmo determinando adequadamente o parâmetro μ para o tamanho dos padrões temporais, existirá ambigüidade entre todos aqueles que apresentarem mesma média móvel. A figura 2.11 ilustra quatro padrões temporais de tamanho e forma distintos que ao final, obtêm a mesma representação temporal (valor 0,6) em uma memória exponencial de $\mu = 0{,}9$.

As memórias podem ser analisadas também de acordo com profundidade e resolução. A profundidade indica o quão distante no tempo a memória pode representar informação, e a resolução indica a qualidade da informação armazenada para cada instante de tempo passado. Por estes critérios a memória de atraso tem pequena

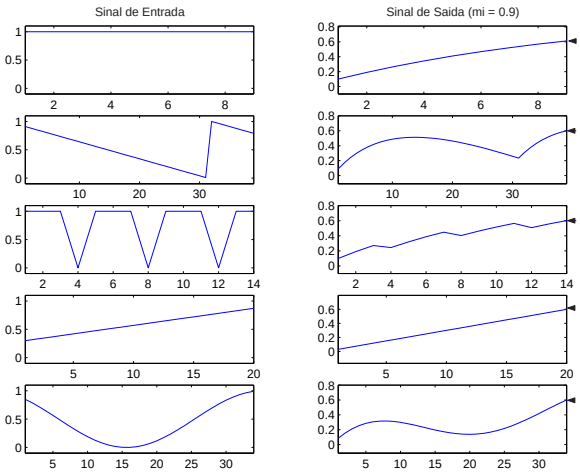

Figura 2.11: Exemplo de ambigüidade de representação em memória exponencial para diferentes sinais de entrada.

profundidade e alta resolução, pois armazena um único valor do passado sem perda de informação, e a memória exponencial tem grande profundidade e pequena resolução, pois representa teoricamente todos os dados do passado mas com perda exponencial na qualidade da informação.

As memórias Gama, generalizam o comportamento das memórias de atraso e das memórias exponenciais. Portanto podem ser criadas num contínuo de variações entre memórias de alta a baixa profundidade e resolução (MOZER, 1993). No entanto, uma única memória Gama é composta por mais de uma memória exponencial e aplicações práticas desta memória exigem o uso de linhas de memórias Gama. O custo computacional alto e a quantidade de variações de parâmetros possíveis dificultam a aplicação do modelo.

As unidades de Habituação aparentemente têm um comportamento parecido com o das memórias exponenciais, integrando a informação temporal. No entanto essa integração tem um viés para a acumulação do sinal de entrada até seu ponto de saturação. A saída da unidade reflete uma noção de quanto tempo atrás a entrada foi alimentada com sinais diferentes de zero. Se o sinal de entrada for 1, a saída da unidade rapidamente chega ao ponto de saturação ou habituação (valor 0). No momento que o sinal de entrada voltar a zero, a unidade lentamente volta ao seu estado de não habituação (valor 1).

A utilidade desta memória foi comprovada com um experimento teórico para reconhecimento de padrões espaço-temporais chamados de Sonogramas de Banzhaf (STILES; GHOSH, 1995). Neste experimento, cada classe de padrão espaço-temporal possui 30 dimensões espaciais e duração de 40 instantes de tempo. A figura 2.12 resume todas as classes básicas. O tempo está representado pela coordenada x e o espaço, na coordenada y. A variação de tonalidade entre branco e preto reflete valores contínuos de 0 a 1 respectivamente.

Para dificultar o problema, as seqüências espaço-temporais utilizadas no treinamento e teste são geradas com adição de ruído e perturbações de rotação, magnitude e translação das classes básicas, como ilustrado na figura 2.13.

O autor utiliza 30 unidades de habituação na entrada de um *Perceptron de Múltiplas Camadas* ou MLP (do inglês *MultiLayer Perceptron*) (figura 2.14). Cada unidade é responsável por uma dimensão espacial do padrão espaço-temporal.

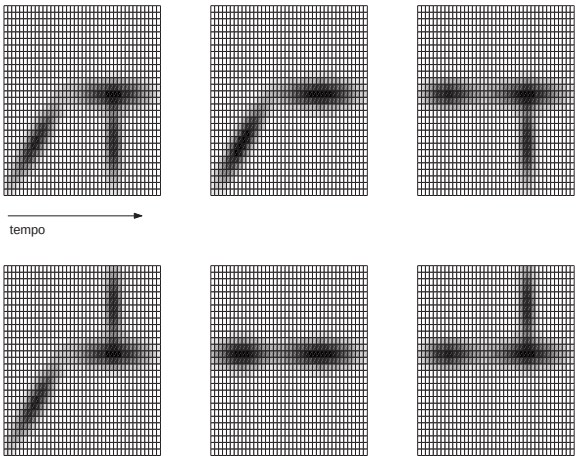

Figura 2.12: Classes básicas (sonogramas de Banzhaf) utilizadas no experimento com Unidades de Habituação.

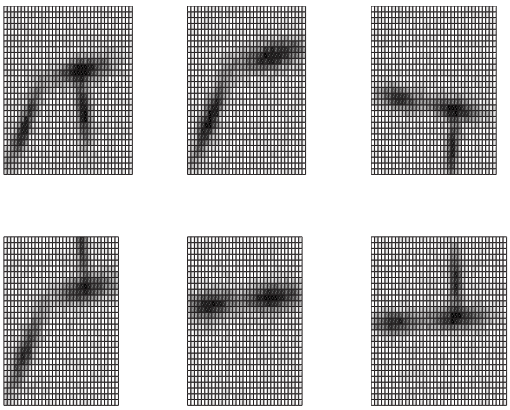

Figura 2.13: Exemplo de sinais gerados a partir de deformações nas classes básicas.

A rede é treinada para classificar corretamente todas as seqüências temporais após a apresentação completa de cada uma delas. Como a rede neural não possui conexões retroalimentadas, nem linhas de atraso, a memória que dispõe provém unicamente da representação gerada pelas unidades de habituação, ou seja, se ao final de uma seqüência temporal, for necessário ter acesso a um ou mais dados de entrada anteriores, esta informação deverá estar disponível nas unidades de habituação. De fato, o conjunto das unidades de habituação gera uma representação espacial não ambígua para cada padrão espaço-temporal utilizado. Para comprovar, gerou-se imagens representando a saída das unidades de habituação resultantes da aplicação de algumas seqüências espaço-temporais (figura 2.15). Foram utilizados os mesmos parâmetros relatados pelo autor ($\alpha = 0{,}07$ e $\tau = 0{,}7$). É importante notar, no entanto, que seqüências semelhantes podem gerar uma representação final muito próxima, como o caso das classes A e C.

Pela figura 2.15 fica evidente que o problema foi resolvido porque as seqüências temporais utilizadas não são demasiado longas, nem ambíguas. No entanto estes dados teóricos foram criados para simular seqüências reais, como fluxos sensório-motores de robôs. Estes, por outro lado, são tipicamente ruidosos, ambíguos e de longa duração.

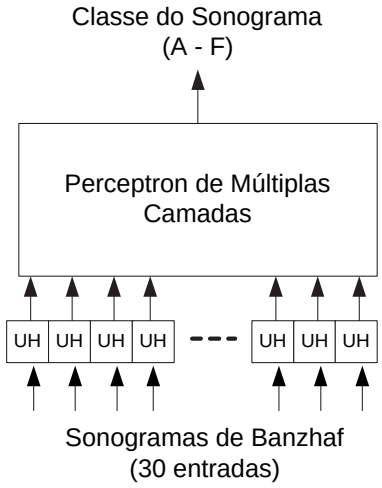

Figura 2.14: Unidades de habituação (UH) aplicadas a um *Perceptron de Múltiplas Camadas*.

Cabe ressaltar que este mesmo problema não seria resolvido se ao invés de utilizar as unidades de habituação, fossem aplicadas memórias exponenciais. Para que a memória exponencial propague informação por longos períodos, sua taxa de decaimento deve ser baixa. Mas esta mesma taxa, determina a velocidade de memorização. Logo, eventos rápidos, como os traços verticais das classes C e F da figura 2.15 dificilmente se manteriam memorizados até o momento de classificação, gerando representações ambíguas ao final das seqüências.

É importante perceber o viés de acumulação de atividade das unidades de Habituação. Estas unidades propagam no tempo sinais de entrada diferentes de 0. Se os valores dos padrões temporais do experimento anterior forem invertidos, a representação das unidades de Habituação não será eficaz para a tarefa de classificação (figura 2.16). As pequenas regiões de sinais em "zero" não são propagadas até o momento de classificação e todas as seqüências espaço-temporais acabam por ter a mesma representação final.

Para finalizar esta comparação de modelos de memória. Gerou-se um sinal temporal simples e o mesmo foi aplicado aos quatro tipos de memórias apresentados neste capítulo. Os resultados podem ser vistos na figura 2.17. Foi utilizada uma linha de 3 memórias de atraso unitário, uma memória exponencial de $\mu = 0{,}9$, uma memória Gama de $\omega = 6$ e $\mu = 0{,}4$ e uma Unidade de Habituação de $\tau = 0{,}7$ e $\alpha = 0{,}07$. As setas indicam exemplos de instantes onde houve ambigüidade de representação na saída de cada tipo de memória. Na maioria dos casos, estes mesmos instantes não são ambíguos para as demais memórias (apresentam valores diferentes). Este talvez seja um indício de que a representação de sinais temporais possa ser feita adequadamente com o uso de diferentes tipos de memória, cada um, ressaltando uma propriedade temporal distinta.

2.2 Arquiteturas Baseadas em Mapas Auto-organizáveis

> *"Dentre as arquiteturas e algoritmos sugeridos para redes de neurônios artificiais, o mapa auto-organizável tem a propriedade especial de criar representações internas espacialmente organizadas das diversas características dos sinais de entrada"* (KOHONEN, 1990).

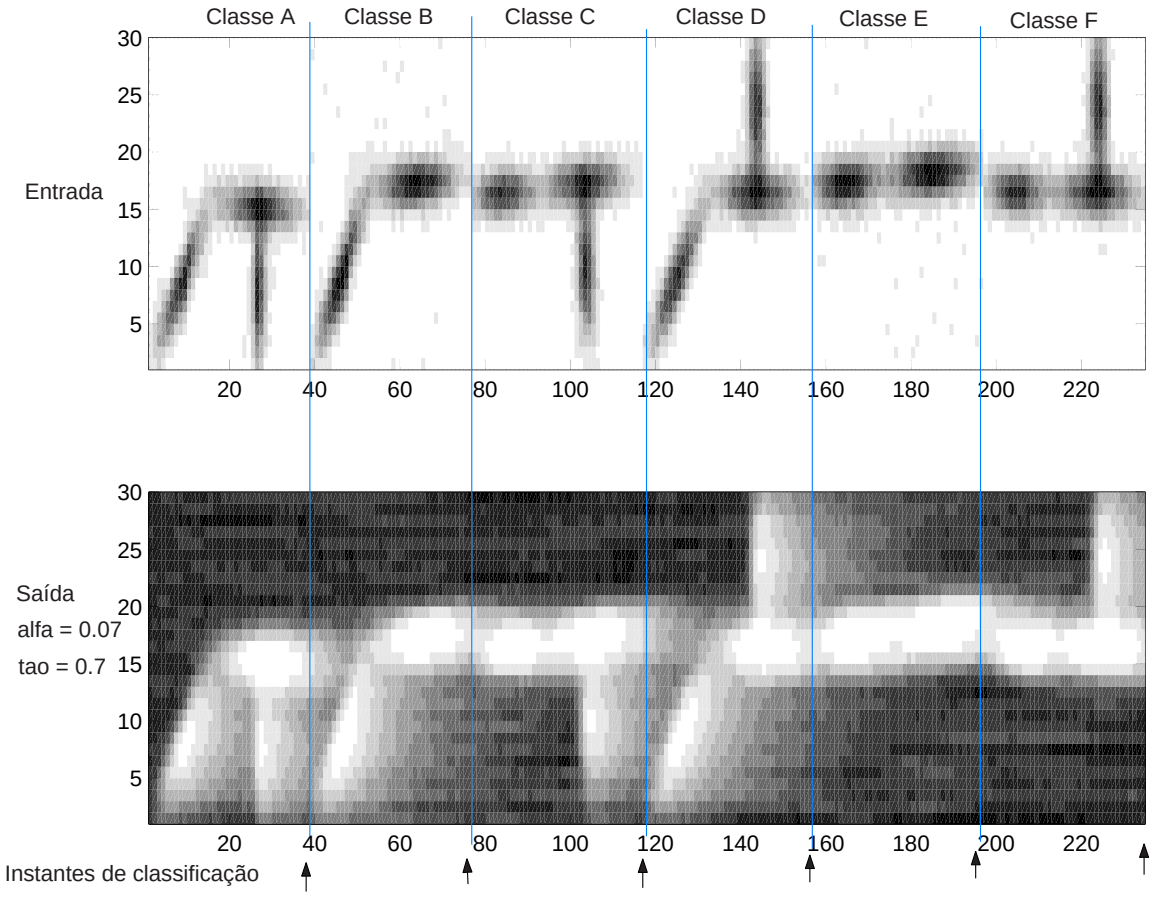

Figura 2.15: Saída dos neurônios de habituação aplicados a uma seqüência de sonogramas de Banzhaf.

Os mapas auto-organizáveis ou SOMs (do inglês, *Self-Organizing Map*) são arquiteturas conexionistas de aprendizado não-supervisionado que realizam quantização vetorial sobre dados tipicamente espaciais. A arquitetura é composta por uma camada de M neurônios protótipos distribuídos num mapa uni ou bidimensional. Os neurônios possuem portanto informação espacial, ou seja, um vetor de posição relativa ao mapa. Esta informação é essencial para o cálculo de vizinhança utilizado na aprendizagem dos neurônios (KOHONEN, 1990).

Os mapas auto-organizáveis são redes competitivas. Para cada padrão de entrada $\mathbf{x}$, é selecionado o neurônio vencedor v, cujos pesos sinápticos $\mathbf{w_v}$ mais se aproximam do padrão. Geralmente utiliza-se a distância euclidiana para calcular a similaridade entre o padrão de entrada e os pesos dos neurônios protótipos. O neurônio vencedor é determinado pela equação 2.10.

$$v = argmin_{1 \leq i \leq M}\{||\mathbf{x} - \mathbf{w_i}||\} \tag{2.10}$$

Uma vez determinado o vencedor, é realizada a adaptação sináptica sobre os pesos de todos os neurônios ($\mathbf{w_i}$) pela equação 2.11, que basicamente faz com que o neurônio vencedor e seus vizinhos se tornem mais especializados no reconhecimento do último padrão de entrada apresentado ($\mathbf{x(t)}$).

$$\mathbf{w_i}(t+1) = \mathbf{w_i}(t) + h_{vi}(t)[\mathbf{x(t)} - \mathbf{w_i}(t)] \tag{2.11}$$

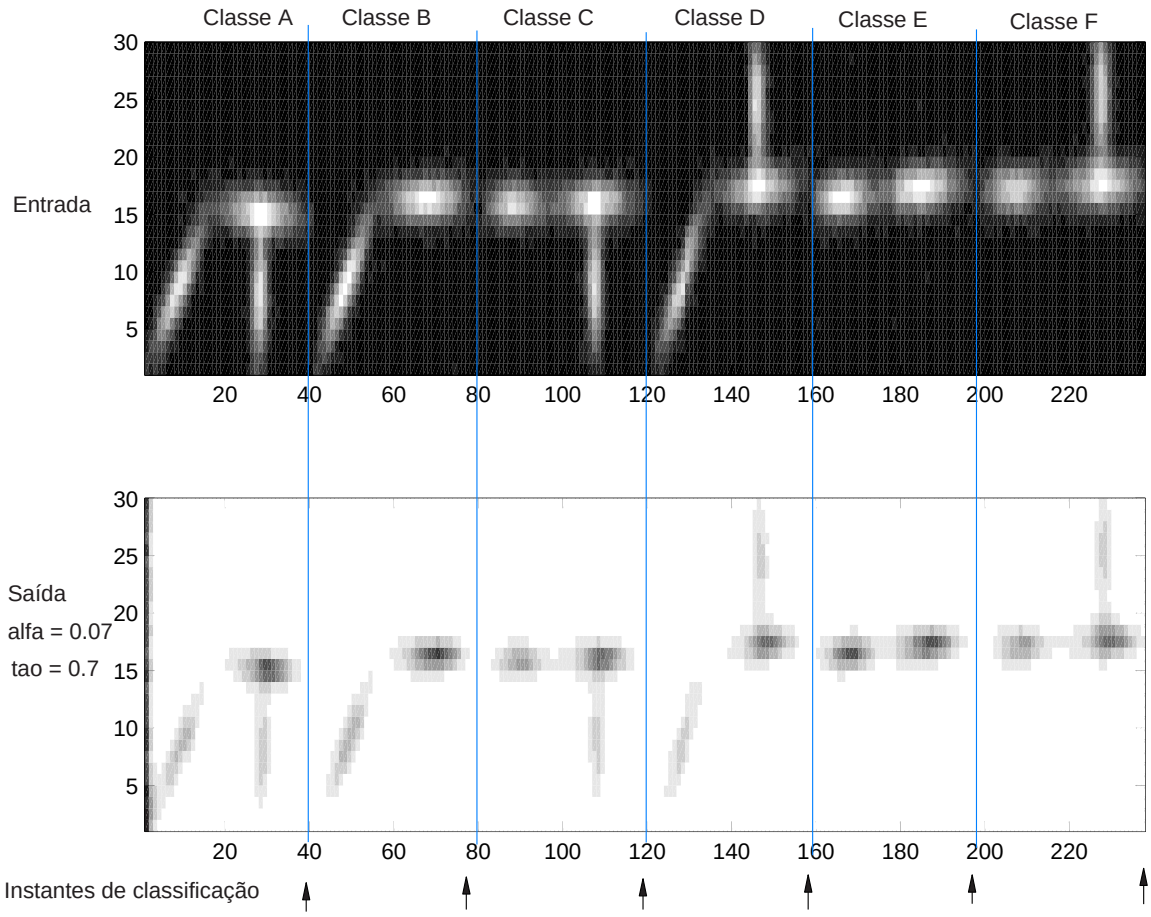

Figura 2.16: Saída dos neurônios de habituação aplicados a seqüência de sonogramas de Banzhaf invertidos.

A determinação da vizinhança é feita em geral com uma função contínua $h_{vi}(t)$ em forma de "sino" com largura σ. Esta função (equação 2.12), utiliza a distância da posição do neurônio vencedor ($\mathbf{r_v}$) em relação à posição dos demais neurônios ($\mathbf{r_i}$) no mapa auto-organizável. O valor final é multiplicado pela taxa de aprendizado $\delta(t)$ que pode ser uma constante ou uma função que decai linearmente no tempo.

$$h_{vi}(t) = \delta(t) e^{\frac{-||\mathbf{r_i}-\mathbf{r_v}||^2}{\sigma^2}} \tag{2.12}$$

Neurônios próximos tendem a reconhecer características similares e portanto, pode-se dizer que o mapa de neurônios preserva a topologia dos dados de entrada. Os neurônios protótipos se organizam conforme a distribuição estatística dos dados, correspondendo a uma média que minimiza o erro de recuperação dos dados de entrada (esta é exatamente a noção de quantização vetorial).

A figura 2.18 representa graficamente os elementos básicos de um SOM. A saída de um SOM pode ser o índice do protótipo vencedor ou um vetor binário com todos os bits em zero, exceto o bit que corresponde ao protótipo vencedor.

A representação gerada pelos mapas auto-organizáveis é tipicamente espacial, pois a saída se refere unicamente ao último padrão de entrada apresentado. A figura 2.19 apresenta uma seqüência temporal de duas dimensões em forma de "8" gerada a partir das equações 2.13 e 2.14 para $1 \leq t \leq 200$. A seqüência temporal pode ser interpretada como a trajetória de um robô realizada dentro de um ambiente.

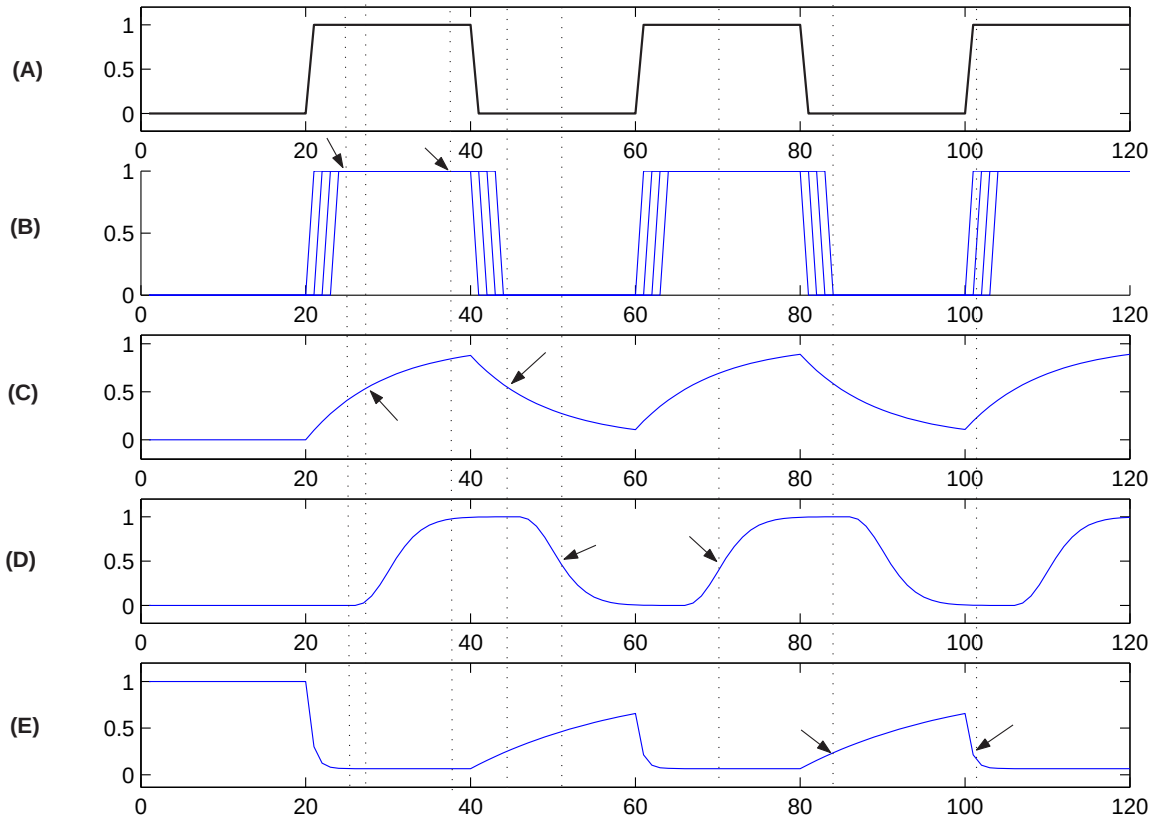

Figura 2.17: (A) Sinal temporal de entrada. (B) Respostas geradas por uma linha de atrasos unitários de 3 neurônios (4 sinais sobrepostos). (C) Resposta de uma memória exponencial de $\mu = 0{,}9$. (D) Resposta de uma memória gama de $\omega = 6$ e $\mu = 0{,}4$. (E) Resposta de uma unidade de habituação de $\alpha = 0{,}07$ e $\tau = 0{,}7$.

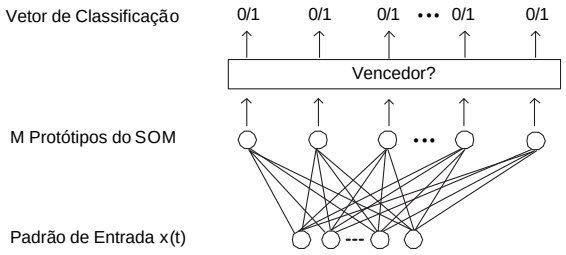

Figura 2.18: Modelo dos Mapas Auto-Organizáveis.

Os 200 pontos obtidos são submetidos a um SOM com o objetivo de segmentar a seqüência temporal em trechos significativos. Foi utilizado para isso, um SOM unidimensional circular de 20 protótipos com parâmetros $\delta = 1$ e $\sigma = 1{,}5$. Para aumentar a convergência e estabilidade dos resultados, δ e σ são decrementados por constantes a cada apresentação da seqüência temporal, de forma que no final do treinamento resultem respectivamente em 0 e 1.

$$x(t) = 0{,}5sin(\frac{t}{200}4\pi) + 0{,}5 \tag{2.13}$$

$$y(t) = 0{,}5cos(\frac{t}{200}2\pi) + 0{,}5 \tag{2.14}$$

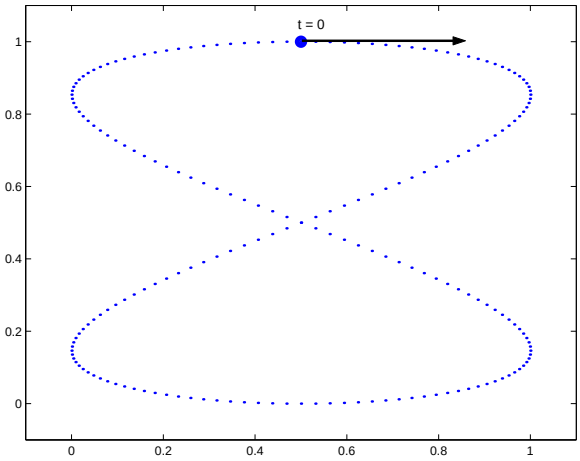

Figura 2.19: Seqüência temporal em forma de "8".

O resultado após 10 apresentações da seqüência é ilustrado na figura 2.20. A saída do SOM para cada instante de tempo da seqüência temporal é apresentada no gráfico inferior e no superior são apresentados os valores finais dos pesos dos neurônios protótipos, representados pelos círculos pretos. A ligação entre eles representa a vizinhança dos neurônios no mapa auto-organizável. A vizinhança é um indicativo de semelhança de padrões. Sob este aspecto, o SOM padrão gerou uma vizinhança correta, no entanto, ela não reflete a seqüência temporal adequadamente.

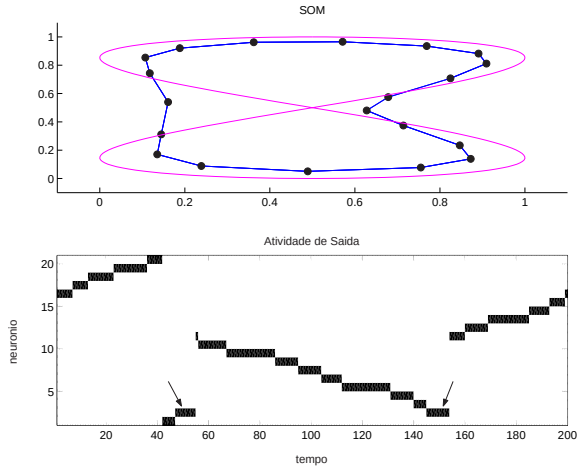

Figura 2.20: Superior: resultado dos protótipos do SOM após 10 apresentações da seqüência temporal. Inferior: saída do SOM para cada ponto da seqüência temporal.

A seqüência temporal escolhida é um caso muito simples, pois possui somente um ponto em que é ambígua, localizado na coordenada (0,5;0,5). Os dois instantes em que a seqüência passa por este ponto são $t = 50$ e $t = 150$, indicados pelas setas no gráfico inferior da figura 2.20. O gráfico indica que o SOM classifica os dois instantes com o mesmo neurônio protótipo. A razão disto é a falta de memória do SOM. Os demais modelos apresentados nesta seção são alterações sobre o SOM com adição de algum tipo de memória para que a atividade da saída (neurônios vencedores) reflita tanto propriedades espaciais como temporais e distinga casos de ambigüidade espacial como este. A mesma seqüência temporal é submetida a todos os modelos para comparar a representação gerada. As condições de treinamento são

mantidas para simplificar a comparação dos modelos.

2.2.1 SOM com Centro de Atenção

O modelo proposto em (GOPPERT; ROSENSTIEL, 1995) reproduz a atividade continuamente variável dos neurônios naturais, fazendo com que os neurônios protótipos tenham suas atividades incrementadas ou decrementadas suavemente. Para tal, o modelo altera o método de seleção do protótipo vencedor, incluindo um fator de "Distância de Atenção", correspondente ao segundo fator do lado direito da equação 2.15. O modelo é apresentado na figura 2.21.

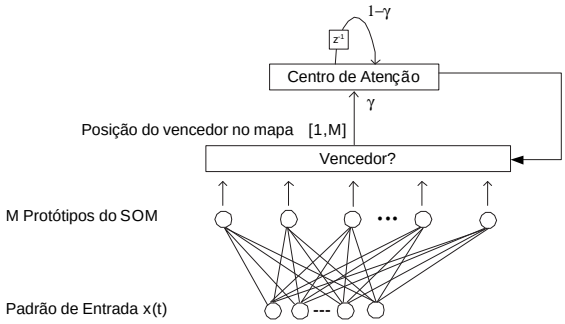

Figura 2.21: Modelo de Mapa Auto-Organizável com Centro de Atenção

$$v = argmin_{1 \leq i \leq M}\{||\mathbf{x} - \mathbf{w_i}||(1 + \frac{||\mathbf{C_A} - \mathbf{r_i}||^2}{\theta^2}))\} \tag{2.15}$$

A constante θ representa o tamanho da região de atenção e determina a importância da distância sensorial em relação à distância de atenção. O$\mathbf{C_A}$ é denominado o centro de atenção e seu domínio é o mesmo dos vetores de posição dos neurônios protótipos ($\mathbf{r_i}$). O centro de atenção move-se a cada instante em direção ao neurônio vencedor com uma taxa de velocidade γ, como definido na equação 2.16. Este centro de atenção é determinado pela atividade da rede no passado, mas poderia ser também afetado por processos cognitivos de mais alto nível, gerando um tipo de treinamento supervisionado.

$$\mathbf{C_A}(t+1) = (1-\gamma)\mathbf{C_A}(t) + \gamma\mathbf{r_v} \tag{2.16}$$

O neurônio vencedor será aquele que tiver seus pesos mais similares ao padrão de entrada e sua posição no mapa de neurônios for próxima ao centro de atenção. Estas duas restrições unidas com o fato do centro de atenção se mover a uma velocidade limitada, fazem com que neurônios próximos no mapa auto-organizável representem características similares e correlacionadas no tempo.

Os resultados da seqüência "8" sobre esta variação de SOM podem ser vistos na figura 2.22. Foram utilizados as mesmas condições de treinamento com os parâmetros adicionais $\theta = 2$ decrementado linearmente até 0,3 e $\gamma = 0{,}1$ decrementado linearmente até 0,01. O SOM com centro de atenção utiliza a proximidade temporal no cálculo de semelhança de padrão, e por isso, gerou uma organização no mapa de forma que a vizinhança dos neurônios refletiu a seqüência temporal. Os instantes ($t = 50$ e 150) em que a seqüência passa pelo ponto de cruzamento são representados por dois protótipos diferentes (10 e 1), como pode ser observado no gráfico inferior da figura 2.22.

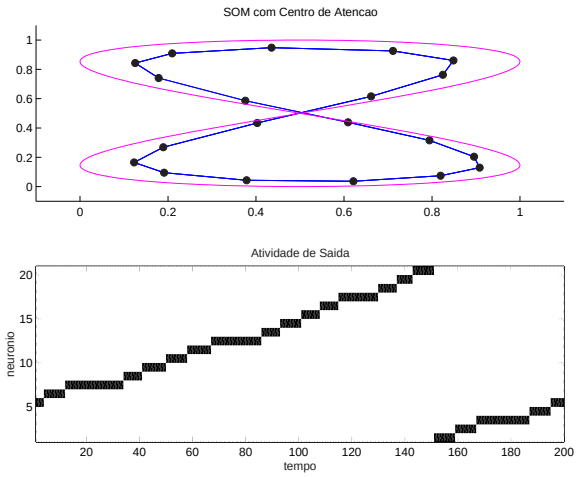

Figura 2.22: Superior: pesos finais dos neurônios do SOM com centro de atenção. Inferior: saída do SOM com centro de atenção pela aplicação da seqüência temporal.

Este resultado condiz com o obtido pelo autor, no entanto, correspondem a somente 30% dos casos realizados num total de 30 experimentos. Nos demais casos, se obteve resultados semelhantes aos do SOM tradicional. A representação temporal final é sensível aos pesos iniciais dos neurônios protótipos.

A representação temporal da seqüência, ilustrada no gráfico inferior da figura 2.22 é bastante satisfatória pois, ao contrário do que houve no SOM padrão (figura 2.20), a atividade na saída não sofre saltos abruptos no decorrer da seqüência. Isto significa que o mapa reflete espacialmente a contigüidade temporal da seqüência. Esta propriedade é importante pois garante estabilidade na representação mesmo na existência de perturbações no sinal, como ruído, deslocamentos no eixo x e y, rotação, etc. A invariância é também positiva para a generalização de conceitos, pelo uso de arquiteturas conexionistas associadas à representação temporal gerada. Cabe ressaltar que a transição entre o neurônio 20 e o neurônio 1 ocorrida no instante $t = 150$ não corresponde a um salto abrupto, pois o mapa auto-organizável é do tipo circular, ou seja, o último neurônio é considerado vizinho do primeiro.

2.2.2 SOMTAD

O SOMTAD (do inglês, *SOM with Temporal Activity Diffusion*) utiliza o conceito de difusão temporal de atividade no mapa auto-organizável para criar memória espaço-temporal (EULIANO; PRINCIPE, 1996). A difusão temporal de atividade é um conceito inspirado nas equações de reação-difusão originalmente propostas por Turing e tipicamente usadas para explicar a formação de padrões naturais (EULIANO; PRINCIPE, 1999).

O resultado final da topologia do mapa do SOMTAD, são neurônios vizinhos que têm correlação espacial e temporal. O autor propôs a aplicação de difusão temporal de atividade em mapas unidimensionais da seguinte forma. Uma vez que um neurônio do mapa seja eleito vencedor, permanece ativo por alguns instantes de tempo, e enquanto isso, facilita a ativação do neurônio vizinho a sua direita. A transmissão de atividade dos neurônios pode ser compreendida pela figura 2.23, onde é apresentado graficamente todo o modelo.

Pela figura, pode-se identificar claramente o uso de memórias exponenciais en-

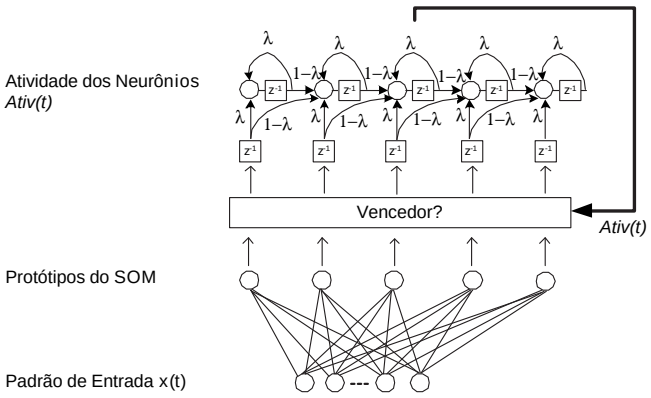

Figura 2.23: Modelo do SOM com difusão temporal de atividade (SOMTAD)

cadeadas na camada de atividade dos neurônios $Ativ(t)$. De fato, o próprio autor ressaltou que o funcionamento é muito semelhante ao das memórias Gama. A única diferença está nas entradas intermediárias que ocorrem em cada memória exponencial com a informação do vencedor do instante anterior. A equação 2.17 define a difusão temporal de atividade ilustrada na figura 2.23.

$$Ativ_i(t) = \lambda[Ativ_i(t-1) + \delta_{i,v(t-1)}] + (1-\lambda)[Ativ_{i-1}(t-1) + \delta_{i-1,v(t-1)}] \quad (2.17)$$

Onde $1 \leq i \leq M$ é o índice do neurônio no mapa auto-organizável unidimensional de M elementos, δ é o delta de kronecker e $v(t)$ é a função de seleção do neurônio vencedor do mapa, dada pela equação 2.18. Esta função é uma variação da equação original 2.10. A constante β é o parâmetro espaço-temporal que determina a influência da informação temporal sobre a seleção do vencedor. A rede torna-se um SOM tradicional se β for zero.

$$v(t) = argmin_{1 \leq i \leq M}\{||\mathbf{x} - \mathbf{w_i}|| - \beta Ativ_i(t)\} \quad (2.18)$$

Este modelo é semelhante ao anterior. Em ambos, a escolha do vencedor favorece neurônios que são vizinhos aos vencedores mais recentes. No entanto, o primeiro modelo possui somente uma memória, o centro de atenção, que favorece somente uma região do mapa, correspondente ao grupo de neurônios próximos ao centro de atenção. Neste último modelo, cada neurônio possui uma memória de sua atividade anterior. Este registro é usado para determinar a própria atividade e a atividade do neurônio vizinho a sua direita. Neste caso, pode haver mais de uma região favorecida, o que corresponderia a seqüências temporais ambíguas, representadas em locais diferentes do mapa auto-organizável. Esta aparente vantagem é prejudicada pelo fato de que a difusão temporal de atividade, tal como foi definida, tem um único sentido, ou seja, sempre para a direita. Além disso, não foram sugeridas formas de se aplicar a mapas auto-organizáveis bidimensionais. Estas limitações fazem com que os resultados dos dois modelos sejam parecidos, como foi constatado pelos experimentos em MATLAB.

A figura 2.24 ilustra o resultado após o treinamento da seqüência temporal "8". Também se obteve uma organização dos neurônios que segue a seqüência temporal, no entanto, foram obtidas em apenas 46% dos experimentos realizados, comprovando

a sensibilidade à configuração inicial dos pesos de cada neurônio protótipo. Os melhores parâmetros encontrados foram $\lambda = 0{,}6$ e $\beta = 0{,}04$.

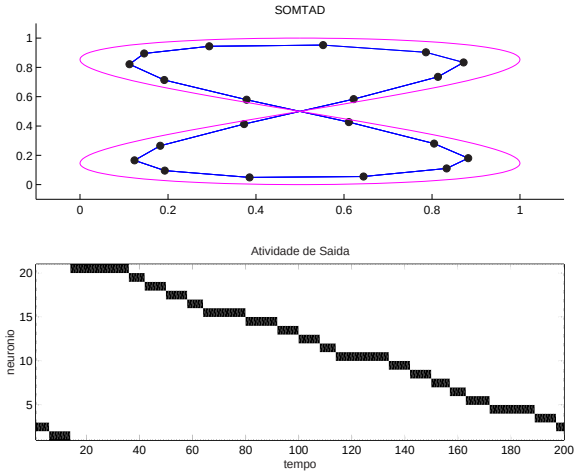

Figura 2.24: Superior: pesos finais dos neurônios do SOMTAD. Inferior: saída do SOMTAD pela aplicação da seqüência temporal.

2.2.3 TKM

O mapa temporal de Kohonen ou TKM (do inglês, *Temporal Kohonen Map*) faz parte de um segundo tipo de mapas auto-organizáveis para tratamento de dados temporais (CHAPPELL; TAYLOR, 1993). Basicamente o TKM realiza a seleção do neurônio vencedor baseado na integração temporal dos erros de quantização dos neurônios, chamada de atividade (figura 2.25). A integração é dada pela equação 2.19, onde i é o índice do neurônio no mapa auto-organizável e d é a constante de retenção de atividade e deve ser um valor entre 0 e 1. O neurônio que apresentar o maior nível de atividade (A) é considerado o vencedor. O neurônio vencedor é aquele cujos pesos mais se aproximam da média móvel dos últimos dados de entrada.

$$A_i(t) = dA_i(t-1) - (1/2)||\mathbf{x}(t) - \mathbf{w_i(t)}|| \tag{2.19}$$

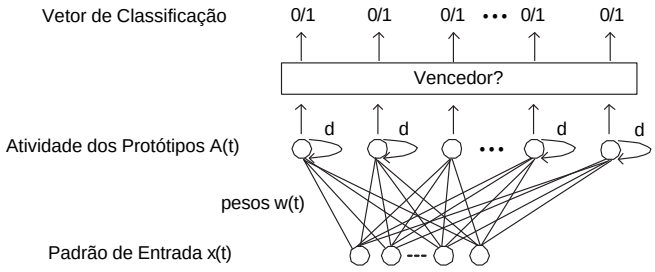

Figura 2.25: Modelo do mapa temporal de Kohonen (TKM)

A operação de integração, como já foi visto nas memórias exponenciais, gera ambigüidades para todas as seqüências com mesma média móvel. Isto explica a existência de ambigüidade observada nos experimentos com a seqüência temporal em forma de "8". A figura 2.26 apresenta um dos resultados obtidos. A taxa de retenção d foi fixada em 0,8 após alguns testes preliminares. Apesar de 23% dos

experimentos resultarem na configuração apresentada no gráfico superior da figura, todas as vezes foi identificado algum tipo de ambigüidade na representação temporal da seqüência, como as apontadas pelo gráfico inferior da figura 2.26. A ambigüidade ocorreu porque logo após o ponto de cruzamento, as médias móveis da entrada se igualam e torna-se impossível distinguir os instantes da seqüência.

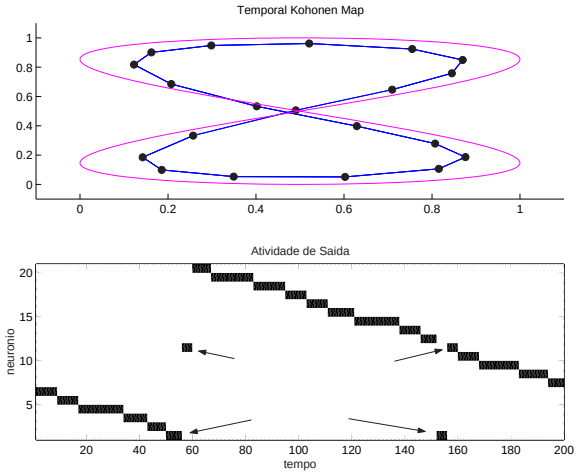

Figura 2.26: Superior: pesos finais dos neurônios do TKM. Inferior: saída do TKM pela aplicação da seqüência temporal.

2.2.4 RSOM

O mapa auto-organizável recorrente ou RSOM (do inglês, *Recurrent Self-Organizing Map*) é uma variação do TKM (VARSTA; HEIKKONEN; R. MILLAN, 1997; KOSKELA et al., 1998). Neste caso, a integração é realizada sobre a diferença entre os valores de entrada e os pesos dos neurônios, como ilustrado na figura 2.27. A integração é dada pela equação 2.20. A informação resultante dá um indicativo de direção do erro para cada neurônio em relação aos últimos dados da seqüência. Portanto, o modelo a utiliza tanto para selecionar o neurônio vencedor, através da equação 2.21, como para realizar as alterações sinápticas (equação 2.22).

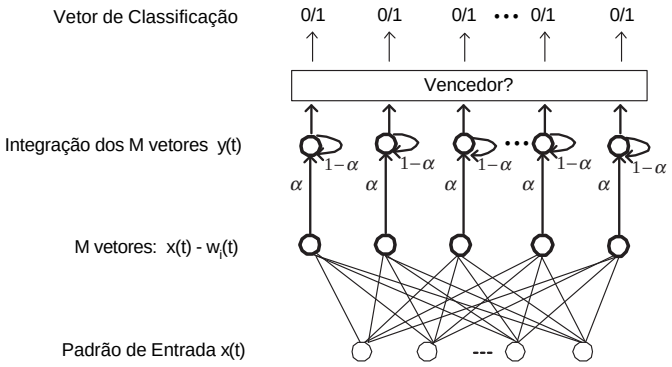

Figura 2.27: Modelo do mapa auto-organizável recorrente (RSOM)

$$\mathbf{y_i}(t) = (1-\alpha)\mathbf{y_i}(t-1) + \alpha(\mathbf{x}(t) - \mathbf{w_i}(t)) \tag{2.20}$$

$$v = argmin_{1 \leq i \leq M} = \{||\mathbf{y_i}||\} \tag{2.21}$$

$$\mathbf{w_i}(t+1) = \mathbf{w_i}(t) + h_{vi}(t)\mathbf{y_i}(t) \tag{2.22}$$

O RSOM é considerado um modelo melhor que o TKM. No entanto, os experimentos com a seqüência temporal "8", revelaram as mesmas ambigüidades encontradas no TKM (figura 2.28). A freqüência com que se obteve a organização dos pesos mostrada no gráfico superior da figura, foi de 30%, levemente superior ao TKM. O melhor valor para o parâmetro α foi de 0,05.

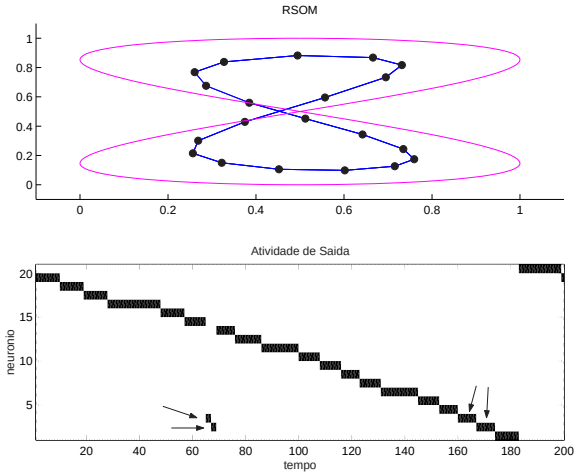

Figura 2.28: Superior: pesos finais dos neurônios do RSOM. Inferior: saída do RSOM pela aplicação da seqüência temporal.

2.2.5 RecSOM

O mapa auto-organizável recursivo ou RecSOM (do inglês *Recursive Self-Organizing Map*) representa um terceiro tipo de solução (VOEGTLIN, 2002). A atividade anterior dos neurônios (dada pelo erro de quantização) é realimentada à sua entrada, como ilustrado na figura 2.29.

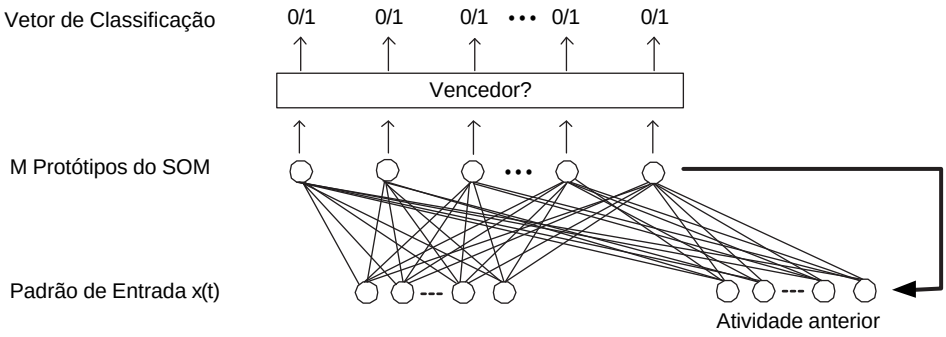

Figura 2.29: Modelo do mapa auto-organizável recursivo (RecSOM)

O erro de quantização é calculado pela equação 2.23, onde se soma a distância sensorial com a distância contextual, balanceadas pelas constantes α e β. Os pesos

dos neurônios protótipos são divididos em dois grupos, $\mathbf{w}^x$ e $\mathbf{w}^y$ relativos aos sinais de entrada e à atividade anterior ou ao contexto, respectivamente.

$$E_i(t) = \alpha||\mathbf{x}(t) - \mathbf{w}_i^x(t)||^2 + \beta||\mathbf{y}(t-1) - \mathbf{w}_i^y(t)||^2 \tag{2.23}$$

A informação do contexto, representada pelo vetor $\mathbf{y}(t)$ é derivada do erro de quantização do instante anterior, como definido pela equação 2.24. Esta função é uma sugestão do autor que permite geração de representações temporais estáveis. O neurônio que apresentar menor erro de quantização é escolhido como vencedor (equação 2.25).

$$y_i(t) = \exp(-E_i(t-1)) \tag{2.24}$$

$$v = argmin_{1 \leq i \leq M} = \{E_i\} \tag{2.25}$$

O espaço de entrada do RecSOM é muito maior que as demais soluções estudadas, por isso, nos testes relatados adiante foram necessárias mais épocas de treinamento. A informação de contexto utilizada pelo modelo faz com que os neurônios se agrupem conforme a semelhança espacial do padrão atual e também conforme a classificação do padrão anterior, que não necessariamente precisa ser semelhante à classificação do padrão atual. Este modelo tem grande potencial de memorização de seqüências temporais de pouca autocorrelação. Provavelmente de grande valia para problemas simbólicos.

Os experimentos do RecSOM com o problema da seqüência temporal "8" apresentaram os melhores resultados, ou seja, com maior freqüência se obteve o resultado esperado da figura 2.30. Em 56% dos 30 experimentos, houve a geração de uma representação estável e não ambígua da seqüência, no entanto foram necessárias 30 apresentações da seqüência para que os pesos estabilizassem. Os melhores parâmetros encontrados foram $\alpha = 1{,}0$ e $\beta = 0{,}5$.

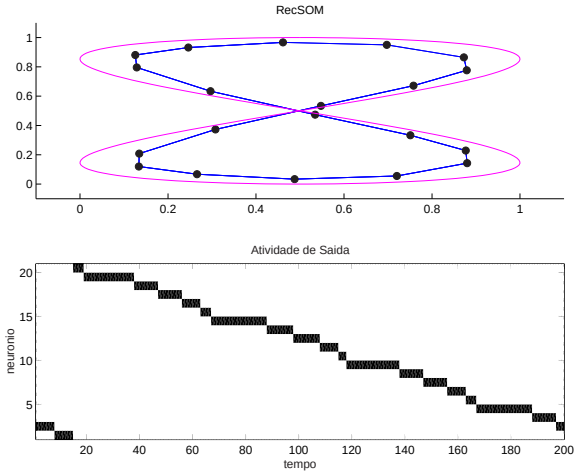

Figura 2.30: Superior: pesos finais dos neurônios do RecSOM. Inferior: saída do RecSOM pela aplicação da seqüência temporal.

2.2.6 SOM-SD

O SOM-SD (do inglês *Self-Organizing Map for Structured Data*) foi originalmente proposto para lidar com dados estruturados, mas pode ser facilmente simplificado para tratar seqüências temporais (HAGENBUCHNER; SPERDUTI, 2003). Este modelo é semelhante ao RecSOM pois o espaço de entrada é aumentado para incluir a informação do último neurônio vencedor, como ilustrado na figura 2.31. A figura mostra um exemplo de mapa unidimensional, portanto, somente um valor (o índice do neurônio no mapa) é apresentado ao mapa auto-organizável. Para mapas bidimensionais seriam utilizados dois valores, correspondentes às coordenadas x e y do neurônio no mapa. A equação 2.26 determina o erro de quantização para a escolha do neurônio vencedor. Trata-se da mesma definição utilizada pelo RecSOM. Somente a natureza da informação contextual é diferente. Neste caso,$c(t-1)$ se refere à posição do neurônio vencedor do instante anterior e $\mathbf{w}^c$ os pesos correspondentes.

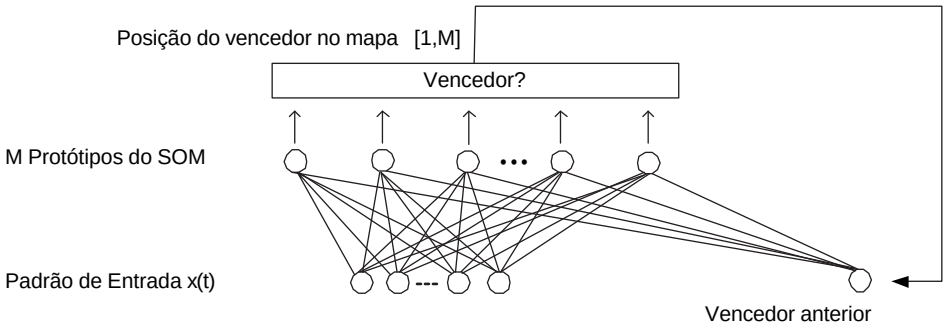

Figura 2.31: Modelo do mapa auto-organizável para dados estruturados (SOM-SD)

$$E_i(t) = \alpha||\mathbf{x}(t) - \mathbf{w}_i^x(t)||^2 + \beta||\mathbf{c}(t-1) - \mathbf{w}_i^c(t)||^2 \tag{2.26}$$

Por ter um espaço de entrada menor que o do RecSOM, se obteve representações estáveis com somente 10 apresentações da seqüência temporal "8". Resultados positivos, como o apresentado na figura 2.32 foram obtidos em 40% dos 30 experimentos realizados, uma taxa levemente inferior ao do RecSOM. Foram utilizados os parâmetros $\alpha = 1{,}0$ e $\beta = 0{,}08$.

2.2.7 Merge-SOM

O Merge-SOM foi o mais recente modelo encontrado na literatura e tem uma abordagem diferente de todos os demais (STRICKERT; HAMMER, 2004). Cada protótipo do SOM possui dois conjuntos de pesos: pesos relativos aos dados de entrada padrão e pesos para reconhecimento do contexto (figura 2.33). Assim como o RecSOM e o SOM-SD, também há o aumento do espaço de entrada com informações contextuais, no entanto, ela é dada pela soma ponderada dos dois tipos de pesos do neurônio vencedor do último instante (equação 2.27). Onde$c$ é o índice do neurônio vencedor e $\mathbf{w}^x$ e $\mathbf{w}^y$ são, respectivamente, os pesos dos neurônios de entrada relativos ao padrão de dados e ao contexto, e β é uma constante que determina a influência dos dois tipos de pesos. O parâmetro β deve ser um valor entre 0 e 1.

$$\mathbf{y}(t) = (1-\beta)\mathbf{w}_c^x(t-1) + \beta\mathbf{w}_c^y(t-1) \tag{2.27}$$

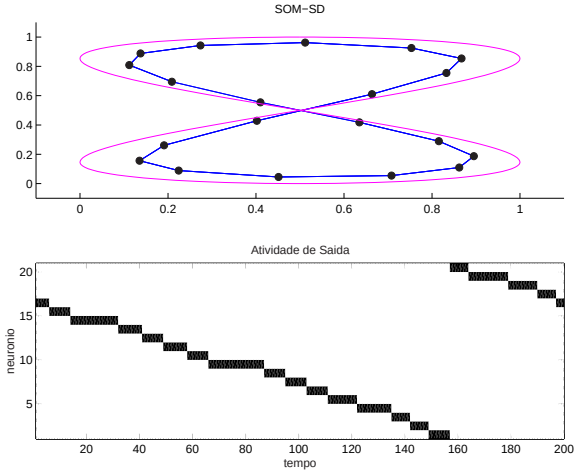

Figura 2.32: Superior: pesos finais dos neurônios do SOM-SD. Inferior: saída do SOM-SD pela aplicação da seqüência temporal.

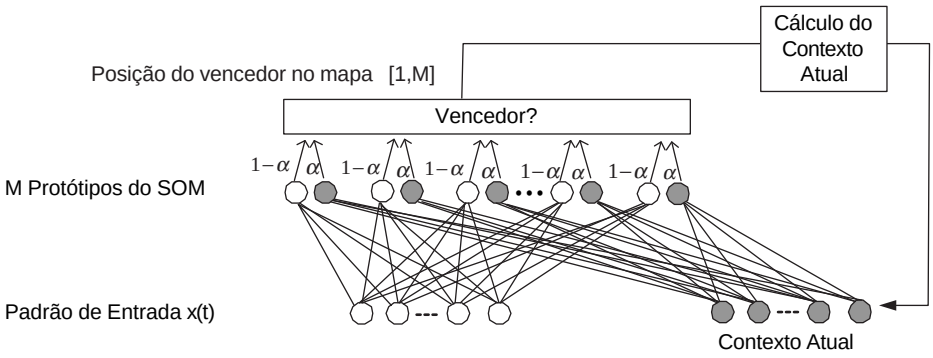

Figura 2.33: Modelo do Merge-SOM

O cálculo do erro de quantização é semelhante ao do RecSOM e do SOM-SD (equação 2.28). A distância euclidiana sensorial é somada com a distância contextual, ponderadas pela constante α, que deve ser um valor entre 0 e 1.

$$E_i(t) = (1-\alpha)||\mathbf{x}(t) - \mathbf{w}_i^x(t)||^2 + \alpha||\mathbf{y}(t-1) - \mathbf{w}_i^y(t)||^2 \qquad (2.28)$$

Os resultados obtidos nos experimentos da seqüência temporal "8" foram pouco satisfatórios. Se obteve a organização dos pesos esperada em somente 23% dos casos (gráfico superior da figura 2.34). Além disso, houve ambigüidade na representação temporal em 100% dos casos, como pode ser visto pelo gráfico inferior da figura 2.34. Estes resultados correspondem aos parâmetros $\alpha = 0{,}5$ e $\beta = 0{,}5$, selecionados após alguns testes preliminares.

2.2.8 SARDNET

Este interessante modelo, denominado SARDNET (do inglês, *Sequential Activation Retention and Decay NETwork*), gera representação esparsa e contínua de seqüências temporais a partir de traços de atividades de um mapa auto-organizável padrão (JAMES; MIIKKULAINEN, 1995). O modelo é ilustrado na figura 2.35. Os sinais de entrada alimentam um mapa auto-organizável. O protótipo vencedor do mapa leva a atividade do neurônio de saída correspondente ao valor 1. A cada

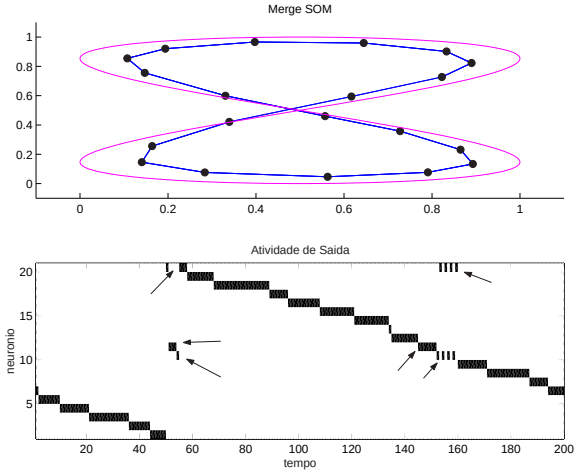

Figura 2.34: Superior: pesos finais dos neurônios do Merge-SOM. Inferior: saída do Merge-SOM pela aplicação da seqüência temporal.

instante de tempo, todos os neurônios de saída sofrem decaimento constante na atividade. A representação de uma seqüência temporal é dada pela configuração de todos os neurônios de saída.

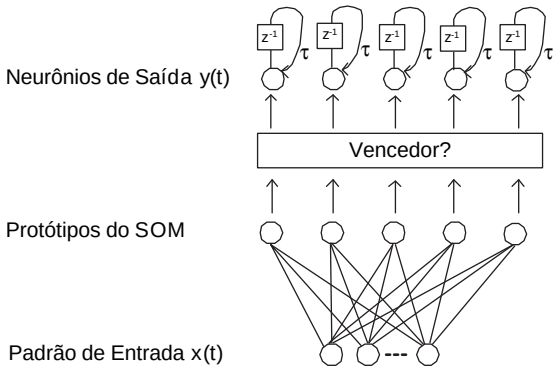

Figura 2.35: Modelo SARDNET

A equação 2.29, rege o decaimento de atividade dos neurônios de saída da rede. Onde i é o índice do neurônio na camada de saída, τ é a taxa de decaimento de atividade e $0 \leq \tau \leq 1$.

$$y_i(t) = \tau y_i(t-1) \tag{2.29}$$

Para que seqüências temporais repetitivas não gerem ambigüidade na representação, o mapa auto-organizável impede que o protótipo vencedor se repita de um instante para o seguinte. Esta restrição faz com que o mapa possa ter dois protótipos representando o mesmo padrão de entrada.

O autor testou o modelo com seqüências temporais simbólicas e obteve bons resultados. A seqüência temporal "8" foi aplicada ao modelo com parâmetro $\tau = 0{,}95$. Após estabilizar a representação do SOM com a apresentação da seqüência em 10 épocas, obteve-se a organização ilustrada no gráfico superior da figura 2.36, exatamente como o SOM. A atividade dos neurônios de saída nos dois instantes em que a seqüência passa pelo ponto de cruzamento $(0{,}5; 0{,}5)$ são diferentes, como se

pode observar pelo gráfico inferior da mesma figura. Claramente a representação distingue todos os instantes da seqüência temporal, inclusive a situação ambígua.

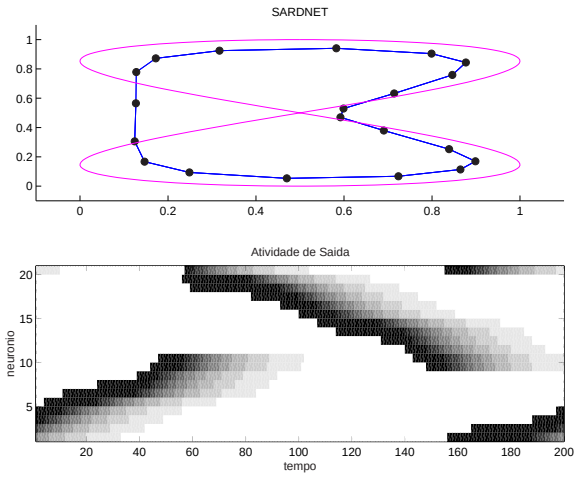

Figura 2.36: Superior: pesos finais dos neurônios do SARDNET. Inferior: saída do SARDNET pela aplicação da seqüência temporal.

A capacidade de representação temporal do SARDNET é muito maior que as outras arquiteturas baseadas em mapas auto-organizáveis. Enquanto que nas outras arquiteturas, cada protótipo deve ser alocado para uma seqüência espaço-temporal, nesta, pelo fato de adotar a representação esparsa, a quantidade de seqüências temporais representáveis pode ser muito maior que a quantidade de protótipos do SOM e o tamanho das seqüências depende da taxa de decaimento escolhida e do grau de ambigüidade delas.

O SARDNET pode ser visto como um SOM tradicional aplicado a memórias de habituação na saída. O sinal temporal resultante gera um rastro dos últimos neurônios vencedores. Portanto, pode-se inferir que o modelo é particularmente útil para representação de seqüências temporais que não sejam demasiadamente repetitivas, assim como já verificado para as memórias de habituação. O resultado dos testes indicou que a limitação imposta ao SOM neste modelo, impedindo que o mesmo neurônio seja vencedor em dois instantes de tempo consecutivos, aparentemente não soluciona o problema de seqüências repetitivas de domínio contínuo. Provavelmente esta propriedade deva ser útil somente em seqüências temporais discretas, foco principal de pesquisa do criador do modelo.

2.2.9 ARAVQ

O modelo ARAVQ (do inglês, *Adaptive Resource Allocating Vector Quantization*) foi criado para aplicações de agentes autônomos, mais especificamente para o tratamento do fluxo de sinais sensoriais multidimensionais de robôs. O modelo perde em generalidade comparado com as arquiteturas de segmentação tradicionais, mas possui grande potencialidade nas tarefas para as quais foi desenvolvido, pois se baseia em informações implícitas que outros algoritmos não aproveitam, como a dependência ou relação temporal entre os vetores de dados apresentados e a distribuição não uniforme das amostras (LINåKER; NIKLASSON, 2000).

O modelo é orientado para detecção de mudanças nas características do sinal de entrada, e para isso, utiliza o algoritmo de média móvel finita, que codifica a situação

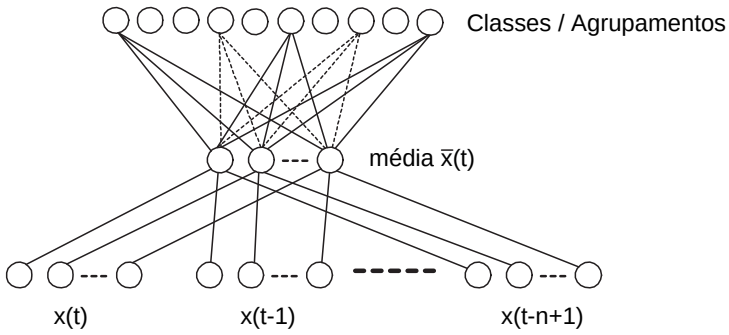

Figura 2.37: Modelo de agrupamento por média móvel (ARAVQ)

atual do robô baseando-se em um intervalo finito de entradas passadas, mantidas em memória. O sinal temporal produzido pela média móvel é usado como entrada para uma rede neural que aplica quantização vetorial incremental, semelhante ao método de alocação de protótipos dos modelos ART (CARPENTER; GROSSBERG, 1987; CARPENTER; GROSSBERG; ROSEN, 1991) (figura 2.37). Portanto, este modelo é bastante distinto dos demais. A quantidade de neurônios protótipos (k) vai aumentando conforme a necessidade até um limite máximo M determinado pelo projetista da rede.

A média móvel é calculada pela equação 2.30, onde o parâmetro n determina o tamanho da janela temporal utilizada e $\mathbf{x}(t)$ é o vetor do padrão de entrada no instante t.

$$\overline{\mathbf{x}}(t) = \frac{1}{n}\sum_{j=0}^{n-1}\mathbf{x}(t-j) \tag{2.30}$$

A partir da janela temporal e da média móvel, são computadas duas medidas: a variância do sinal dentro da janela temporal (equação 2.31) e o erro de quantização do melhor neurônio protótipo existente em relação à janela temporal (equação 2.32).

$$d_{\mathbf{x}(t)} = \frac{1}{n}\sum_{j=0}^{n-1}||\mathbf{x}(t-j) - \overline{\mathbf{x}}(t)|| \tag{2.31}$$

$$d_{\mathbf{w}(t)} = \frac{1}{n}\sum_{j=0}^{n-1}\min_{1\leq l\leq k}\{||\mathbf{x}(t-j) - \mathbf{w_l}(t)||\} \tag{2.32}$$

A inclusão de novos protótipos é realizada se a condição $d_{\mathbf{x}(t)} \leq \min(\epsilon, d_{\mathbf{w}(t)} - \delta)$ for verdadeira. Este teste pode ser interpretado da seguinte forma: a variância do sinal de entrada deve ser menor que a constante mínima de estabilidade (ϵ) e também menor que a distância do protótipo mais próximo existente, reduzido pelo critério de novidade (δ). O novo protótipo recebe o valor da média móvel calculada ($\mathbf{w_{k+1}} = \overline{\mathbf{x}}(t)$) e o número k de protótipos é incrementado. Inicialmente, quando a quantidade de protótipos é zero ($k = 0$), $d_{\mathbf{w}(t)}$ vale $\epsilon + \delta$ para garantir a criação do primeiro protótipo se a variância for menor que o critério de estabilidade.

A equação 2.33 define a seleção do protótipo vencedor, cujos pesos mais se aproximam da média móvel calculada. O aprendizado, diferente dos outros modelos baseados em mapas auto-organizáveis, não leva em conta a vizinhança dos neurônios protótipos. Somente o neurônio vencedor sofre alterações sinápticas. Além

disso, para garantir estabilidade na representação, o aprendizado só é realizado se o erro de quantização for pequeno ($||\overline{\mathbf{x}}(t) - \mathbf{w_v}(t)|| < \frac{\epsilon}{2}$). O aprendizado é dado pela equação 2.34, onde α determina a taxa de aprendizagem.

$$v = argmin_{1 \leq i \leq k}\{||\overline{\mathbf{x}}(t) - \mathbf{w}_i(t)||\} \tag{2.33}$$

$$\mathbf{w_v}(t+1) = \mathbf{w_v}(t) + \alpha[\overline{\mathbf{x}}(t) - \mathbf{w_v}(t)] \tag{2.34}$$

Resumidamente, a arquitetura possui quatro parâmetros que determinam a forma como será segmentada a seqüência temporal. Estes quatro parâmetros devem ser definidos de acordo com as propriedades dos sensores e dos eventos que afetam o robô. O primeiro parâmetro é o tamanho da janela temporal onde é calculada a média móvel (n). O segundo parâmetro é o critério de novidade δ, que determina a criação de novas categorias para representar o estado do robô. O terceiro parâmetro é o critério de estabilidade ϵ, que identifica momentos de transição entre estados, baseado na variância do sinal de entrada. Por último, a taxa de aprendizagem α, que determina a taxa com que os protótipos vencedores se aproximam da média móvel.

Os primeiros experimentos relatados com esta arquitetura utilizaram o robô Khepera com comportamento de *following wall* em um labirinto. Os 8 sinais sensoriais de distância do robô tipicamente ruidosos, foram submetidos ao ARAVQ enquanto o robô explorava o ambiente. A segmentação obtida foi gerada em tempo real. Surgiram protótipos que representaram as configurações básicas encontradas no labirintos, como "corredor", "bifurcação", "curva para direita", etc. (LINåKER; NIKLASSON, 2000).

Cabe ressaltar as seguintes observações sobre este algoritmo:

- O aprendizado é muito rápido e relativamente estável, bastante adequado para aplicações em robótica.
- Diferente dos outros métodos de segmentação, este não descarta eventos raros taxando-os como ruído, pois justamente é voltado a mudanças.
- A segmentação do fluxo sensorial não é feita levando-se em conta a relevância dos eventos para previsão ou qualquer outra aplicação. É um processo não-supervisionado que depende exclusivamente de mudanças nas características do sinal. A representação temporal gerada é do tipo auto-organizada, assim como dos mapas auto-organizáveis já apresentados.
- Utiliza o tempo somente para fins de segmentação e não para representação. Os protótipos gerados correspondem às médias dos padrões de entrada em janelas finitas de tempo e portanto não contêm informação de mudança neste período.

Para comprovar esta última observação, implementou-se em MATLAB uma rede do tipo ARAVQ e aplicou-se à sua entrada a mesma seqüência temporal de duas dimensões em forma de "8" (figura 2.19). Utilizando-se parâmetros adequados no ARAVQ ($\alpha = 0{,}05$, $\delta = 0{,}2$, $\epsilon = 0{,}3$ e $n = 10$), obteve-se a segmentação mostrada na figura 2.38. O importante nesta figura é observar a forma como a seqüência temporal foi segmentada, e não o valor do agrupamento, propriamente dito.

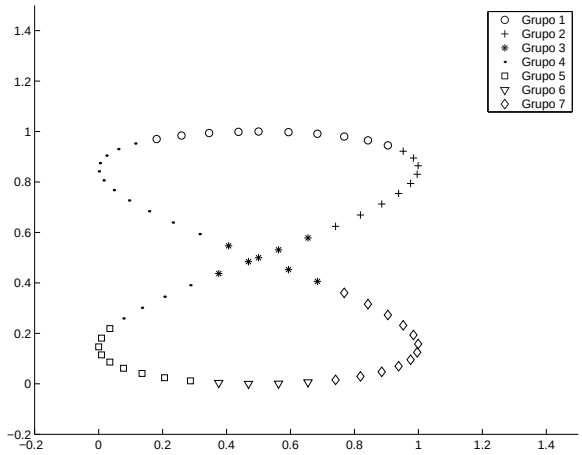

Figura 2.38: Classificação da seqüência temporal pelo ARAVQ.

A segmentação resultante não é satisfatória, pois são classificados da mesma forma (grupo 4), instantes distantes e diferentes da série temporal. O ponto de cruzamento (0,5; 0,5) estudado nas demais arquiteturas também apresenta classificação ambígua (grupo 3). Se for feita uma média dos valores de x e dos valores de y ocorridos próximos aos dois instantes que passam pelo ponto, se verificará que são exatamente iguais. Esta ambigüidade de representação existe independente dos parâmetros escolhidos e a causa é semelhante a identificada no TKM e RSOM. Para que fossem distinguidos os dois trechos que passam pelo ponto (0,5; 0,5), seria necessário se basear na informação de mudança do sinal e não sua média móvel.

2.2.10 Resumo dos Modelos

A tabela 2.2 resume as características principais dos modelos apresentados nesta seção. São elas:

- Descrição sucinta: descreve como a informação temporal é utilizada no modelo.
- Tipo de Representação: define se a representação temporal gerada é de origem simbólica (Local) ou de origem numérica (Esparsa).
- Convergência dos protótipos: indica a freqüência com que se observou a organização esperada dos protótipos para a seqüência temporal "8".
- Ambigüidade na representação: indica se houve ambigüidade na representação temporal para os casos em que se observou a convergência correta dos protótipos.

Como o modelo ARAVQ é um tipo de rede competitiva que não utiliza vizinhança, ou seja, não é baseada em mapas auto-organizáveis, não se verificou a organização espacial dos protótipos e portanto a coluna *Convergência dos protótipos* não foi preenchida.

Tabela 2.2: Resumo dos modelos de representação temporal baseados no SOM.

Nome	Descrição Sucinta	Tipo de Representação	Convergência dos protótipos	Ambigüidade na representação
SOM c/ centro de atenção	Integração da posição do neurônio vencedor no mapa auto-organizável.	Local	30%	Não
SOMTAD	Integração da atividade dos neurônios vencedores com influência de vizinhança.	Local	46%	Não
TKM	Integração do erro de quantização dos neurônios protótipos.	Local	26%	Sim
RSOM	Integração das diferenças entre os protótipos e o padrão de entrada.	Local	30%	Sim
RecSOM	Atividade do mapa no instante anterior alimentada na entrada do SOM.	Local	56%	Não
SOM-SD	Posição do neurônio vencedor do instante anterior alimentado na entrada do SOM.	Local	40%	Não
Merge-SOM	Contexto do instante anterior alimentado na entrada do SOM.	Local	23%	Sim
SARDNET	Integração da atividade dos neurônios vencedores.	Esparsa	100%	Não
ARAVQ	Alocação dinâmica de protótipos com base na média móvel do sinal.	Local		Sim

2.3 Arquiteturas Baseadas em Compressão de História

O princípio de Compressão de História foi sugerido em um breve artigo de Schmidhuber (SCHMIDHUBER, 1991), na tentativa de suprimir as deficiências das redes recorrentes e dos algoritmos clássicos de treinamento (RTRL e BPTT). Ele argumenta que eventos correlacionados muito distantes no tempo dificilmente são identificados pelos métodos tradicionais, pois os erros retropropagados no tempo se extinguem. Além disso, os algoritmos de treinamento tradicionais não tentam focalizar-se em entradas relevantes, mas todas as entradas apresentadas, desperdiçando eficiência e recursos importantes.

Para reduzir a representação de uma seqüência temporal sem perder informação, Schmidhuber propôs a criação de um módulo previsor. Este previsor, que pode ser implementado com uma rede neural recorrente, deve ser treinado para prever os valores da seqüência temporal a partir da entrada atual e de seu próprio estado interno, ou seja, o previsor deve ter memória. Pelo princípio da Compressão de

História, pode-se descrever a seqüência original listando somente os instantes em que o previsor falhou na sua tarefa e o valor real não previsto da seqüência temporal. O restante da seqüência é recuperado usando-se o próprio previsor. Se o previsor for eficiente, esta descrição será mais compacta que a seqüência original. Uma vez que a descrição da seqüência reduziu e mudou de espaço de representação, é possível aplicar mais um previsor, agora sobre a seqüência dos erros do primeiro e obter uma seqüência provavelmente menor. Este princípio pode ser aplicado indefinidamente obtendo uma hierarquia de previsores e uma representação cada vez mais abstrata da seqüência temporal (figura 2.39). À medida que se sobe na hierarquia, eventos mais distantes no tempo são relacionados pelos previsores. Como resultado, é gerada a representação hierárquica de padrões temporais de grande duração.

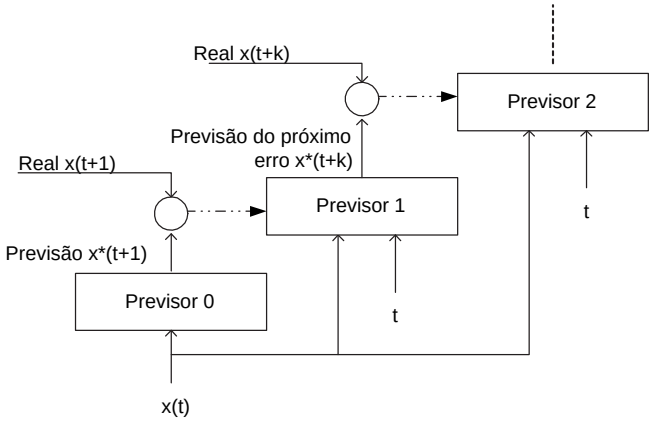

Figura 2.39: Modelo Hierárquico de Compressão de História

O artigo também sugere uma solução com somente dois previsores encadeados, denominados *Automatizador* e *Agrupador* (figura 2.40). O *Automatizador* está no primeiro nível e é treinado para prever sua próxima entrada e o estado interno do *Agrupador*. O *Agrupador* só é ativado quando o *Automatizador* produz um erro de previsão da seqüência temporal. Neste caso, recebe como entrada o valor não previsto pelo *Automatizador* e um identificador único de tempo, usado para seu treinamento. O *Agrupador* é treinado para prever o próximo erro de previsão do *Automatizador*. À medida que o *Agrupador* descobre relações temporais mais globais, o *Automatizador* torna-se apto a aprendê-las pois, como já foi dito, é treinado também para prever o estado interno do *Agrupador*, e ao fazer isto, pode prever os mesmos eventos.

O autor apresenta o conceito de forma genérica, sem se limitar ao uso de redes neurais como módulos previsores. No entanto, as redes neurais são utilizadas em seus experimentos, apresentando resultados animadores, pois resolvem problemas que redes recorrentes tradicionais não conseguem tratar. A maior desvantagem deste modelo é a intolerância a ruídos, pois trabalha com um conceito muito rígido de sucesso de previsão. Não existe a noção de previsão parcial. Estas dificuldades foram abordadas em um segundo artigo (SCHMIDHUBER; MOZER; PRELINGER, 1993), onde Schmidhuber estende o princípio de Compressão de História para seqüências temporais contínuas, mas o assunto não será examinado neste trabalho.

Nas subseções seguintes estão resumidos três dos principais trabalhos encontrados na literatura científica que são baseados neste princípio. Todos eles, foram desenvolvidos por Tani com a colaboração de outros pesquisadores. Seus trabalhos

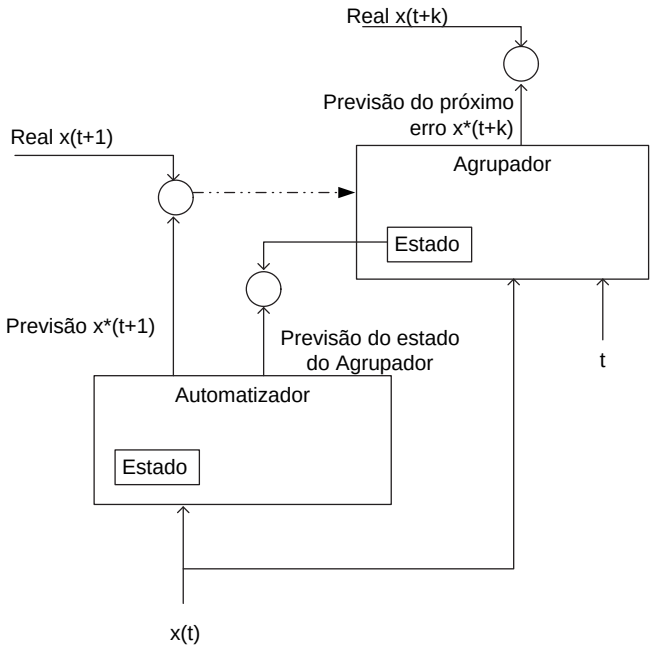

Figura 2.40: Modelo de duas camadas de Compressão de História

estão voltados à criação de representações espaço-temporais para processamento de fluxos sensório-motores de robôs. Todas as arquiteturas sugeridas são treinadas para prever o sinal de entrada, e assim, segmentar o fluxo sensório-motor em representações significativas.

O trabalho de Tani é baseado em sistemas dinâmicos. Ele associa as redes recorrentes aos sistemas dinâmicos, pois elas podem aprender a dinâmica do sistema entrando em sincronia com o mesmo. Uma vez em sincronia, seus estados internos correspondem aos estados do sistema real e portanto podem ser utilizados para previsão ou qualquer outra tarefa que exija a descrição do estado. Em todos os trabalhos citados, uma arquitetura conexionista é criada com o objetivo de segmentar o fluxo sensorial de um robô, sem visar o controle de suas ações. As ações são controladas por sistemas independentes e não possuem aprendizado. Isso porque o objetivo das pesquisas é o de encontrar representações internas úteis. Uma vez obtidas, poder-se-ia utilizá-las para o controle do robô ou para classificar estados importantes durante a interação com o ambiente.

Os modelos descritos a seguir não foram implementados em MATLAB, pois a reprodução dos experimentos seria de grande dificuldade, uma vez que foram utilizados robôs reais em situações particulares. Portanto sua análise se baseia nos próprios comentários dos autores e em observações deduzidas a partir das características conhecidas de cada modelo conexionista utilizado nas soluções.

2.3.1 Modelo de Redes de Elman em Cascata

No primeiro trabalho (NOLFI; TANI, 1999), Tani usa duas redes de Elman em cascata E_1 e E_2 (figura 2.41). E_1 recebe como entrada a leitura dos sensores do robô e é treinada para prever a entrada do próximo instante. A atividade da camada oculta de E_1 alimenta uma rede de segmentação auto-organizável (competitiva), semelhante às arquiteturas ART (CARPENTER; GROSSBERG, 1987). E_2

recebe como entrada o protótipo vencedor da rede de segmentação adicionado de uma entrada que expressa o tempo decorrido desde a última mudança de protótipo. E_2 deve também prever sua próxima entrada, mas funciona em outra escala temporal, pois só propaga quando muda o protótipo vencedor da rede de segmentação. Ambas as redes de Elman são treinadas com o algoritmo de retropropagação de erro incremental padrão. O treinamento é feito em etapas. Primeiramente E_1 é treinado com a seqüência temporal. Depois a rede de segmentação gera protótipos para representar de forma compacta, os estados internos de E_1 com a apresentação da seqüência. Por último E_2 é treinado para prever a seqüência dos protótipos vencedores. O modelo é validado com um experimento real, utilizando o robô Khepera com comportamento fixo de *following wall*.

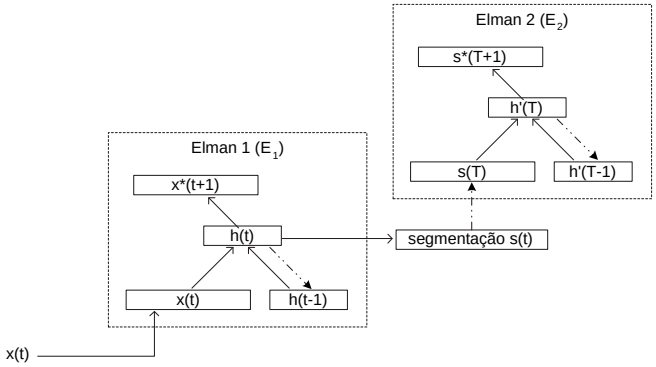

Figura 2.41: Modelo conexionista baseado em Compressão de História usado em (NOLFI; TANI, 1999)

Nos experimentos do autor, a primeira rede de Elman, E_1, representou pequenos trechos da seqüência, gerando classificações como "movendo-se pelo corredor", "virando a direita" ou "virando a esquerda". A rede de segmentação gerou protótipos para cada uma destas situações. A segunda rede de Elman E_2, por ser treinada para prever as mudanças de protótipos, acabou criando representações internas que permitiram a localização do robô no ambiente. A representação temporal resultante se encontra nas camadas ocultas das duas redes de Elman. Cada uma delas representa informações temporais em escalas de tempo diferentes.

Analisando rapidamente este modelo, pode-se identificar dois problemas em potencial. Em primeiro lugar, as redes do modelo são treinadas separadamente, usando as seqüências temporais, no caso, leitura dos sensores do Khepera movendo-se em um labirinto. O aprendizado não é dinâmico e portanto não pode ser aplicado em tarefas como exploração de ambientes. O segundo ponto diz respeito à geração não-supervisionada dos protótipos na rede de segmentação. A segmentação depende de como os neurônios ocultos da rede E_1 se organizam e como as representações dos estados diferem entre si. A criação de protótipos não é realizada levando em conta suas utilidades, portanto, é possível existir protótipos que representem estados inúteis (segmentação muito específica) e também protótipos que representem dois ou mais estados diferentes (segmentação muito genérica). A escolha de parâmetros da rede de segmentação tem um papel fundamental na criação da representação temporal final da rede.

2.3.2 Modelo Hierárquico de Redes de Jordan

Num segundo trabalho (TANI; NOLFI, 1999), o modelo sugerido é inspirado em *Mistura de Especialistas* de Jabobs. O objetivo é segmentar o fluxo contínuo sensório-motor do robô em seqüências úteis para a previsão. A arquitetura é hierárquica (figura 2.42). No primeiro nível, um conjunto de redes neurais recorrentes (redes de Jordan) devem todas prever a leitura dos sensores do próximo instante de tempo. O erro de previsão é calculado comparando a leitura dos sensores no instante $t+1$ com a soma das saídas de todas as redes no instante t, ponderadas por pesos de participação (*gates*), que são atualizados dinamicamente e dependem do grau de acerto de cada rede nos últimos instantes. Os pesos de participação são usados também no treinamento, gerando um tipo de competição entre as redes. Esta competição leva cada rede a se especializar em um determinado trecho do fluxo sensório-motor, representando ao final do treinamento, assim como na primeira arquitetura, situações simples do tipo "movendo-se pelo corredor", etc.

Os níveis superiores da arquitetura hierárquica seguem a mesma estrutura, mas recebem como entrada os pesos de participação (*gate*) da camada inferior e devem prever seus valores no instante $t+k$ ($k >> 1$). O aprendizado é contínuo e utiliza o algoritmo BPTT (do inglês, *Backpropagation Through Time*). O autor utilizou em seus experimentos $k = 10$.

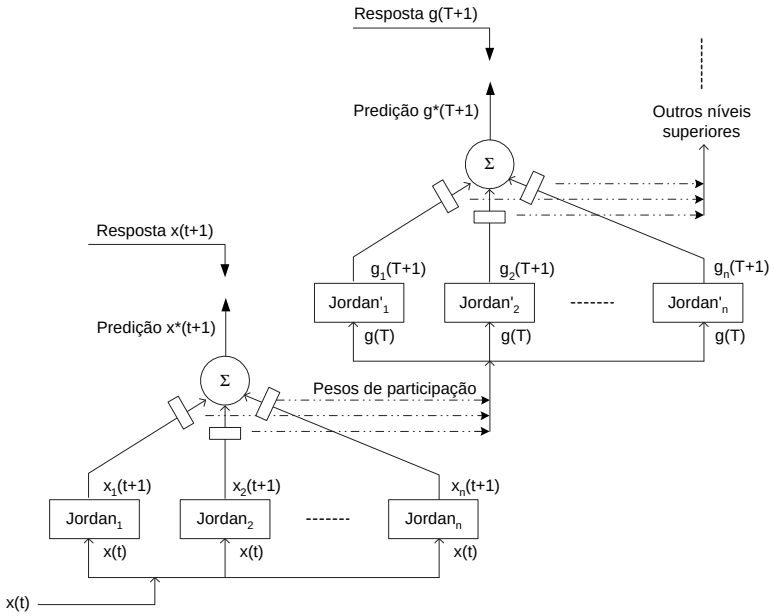

Figura 2.42: Modelo conexionista baseado em Compressão de História usado em (TANI; NOLFI, 1999)

O cálculo do peso de participação é dinâmico, ou seja, computado a cada instante com base no valor anterior, apresentando um comportamento inercial. Como já foi dito, cada rede no primeiro nível é responsável por identificar uma situação. O peso de participação (*gate*) indica qual das situações é a mais provável em cada instante de tempo. Portanto, a combinação de todos os pesos de participação é a representação instantânea do estado da seqüência temporal.

Os experimentos foram realizados com um robô de 20 sensores de proximidade a laser e um comportamento predeterminado de exploração e livre de colisão. Ape-

sar do sucesso nos resultados, tornou-se evidente a limitação quanto ao número de seqüências aprendidas, que depende do número de redes utilizadas.

2.3.3 Modelo de Representações *Top-Down* e *Bottom-Up*

Em seu terceiro trabalho (TANI, 2003), Tani utiliza uma arquitetura hierárquica de dois níveis com influências *bottom-up* e *top-down* (figura 2.43), que deve aprender a realizar seqüências básicas de ações em resposta a situações do ambiente, utilizando um braço mecânico e uma câmera. A arquitetura segmenta o fluxo contínuo sensório-motor em seqüências úteis para previsão, representadas de forma distribuída, diferentemente do artigo anterior, onde o número de seqüências primitivas dependia do número de módulos de redes neurais no primeiro nível. Neste modelo, cada nível possui somente uma rede neural do tipo Jordan. A idéia neste caso é unificar as redes do modelo anterior numa só, criando unidades adicionais na camada de entrada para aumentar o espaço de representação e permitir a coexistência das seqüências na mesma rede. Estas entradas adicionais são as unidades de *biases* paramétricos.

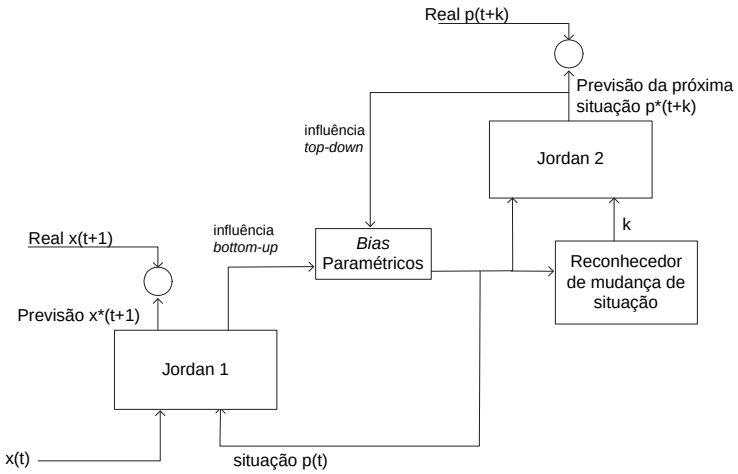

Figura 2.43: Modelo conexionista baseado em Compressão de História usado em (TANI, 2003)

No primeiro nível, a rede recebe os sinais sensoriais, um vetor de *biases* paramétricos e a atividade da camada de saída do instante anterior, característico das redes de Jordan. A rede deve prever a leitura dos sensores do próximo instante. A atividade das unidades de *bias* é calculada juntamente com a aplicação do algoritmo de BPTT na fase de treinamento, onde as seqüências básicas de comportamento são ensinadas.

O *Reconhecedor de mudanças de situação*, ilustrado na figura, representa uma heurística de detecção de mudança na configuração do vetor de *biases* para determinar a ativação da segunda rede de Jordan. A segunda rede é ativada somente após a detecção de mudança na situação, portanto, sua escala temporal é maior, permitindo a representação de seqüências sensório-motoras mais complexas. Esta rede é treinada para prever a próxima configuração do vetor de *bias* ou situação, baseando-se na atividade das unidades de *bias* do primeiro nível, juntamente com o tempo decorrido desde a última mudança de situação.

Terminado o treinamento, realiza-se a simulação. Neste passo, o vetor de *bias*

paramétricos é determinado tanto pelas conexões retroalimentadas da primeira rede, como pela previsão da segunda rede. É utilizado um peso que determina a influência dos *biases top-down* em relação aos *biases bottom-up*. Se a influência *top-down* for maior, o sistema tenderá a manter a seqüência de comportamento aprendido até concluí-la, independente da leitura dos sinais de entrada. Se a influência *bottom-up* for maior, o sistema será bastante sensível às leituras dos sensores, podendo fazer com que o comportamento mude rapidamente para outra seqüência sensório-motora aprendida.

A representação temporal deste modelo encontra-se nos *biases* paramétricos. Estes resumem o estado da seqüência temporal tanto sob o ponto de vista *bottom-up* (seqüência de sinais sensoriais), como do ponto de vista *top-down* (seqüência de ações). É uma representação bastante interessante, mas para sua geração é necessário treinar cada seqüência sensório-motora separadamente e de forma *off-line*. Como o próprio autor ressaltou, dificilmente este modelo poderia apresentar adaptação a novas situações ou comportamentos.

2.4 Visão Geral e Discussão

A análise dos diferentes tipos de representação temporal leva as seguintes conclusões:

- As representações baseadas no princípio de Compressão de História podem descrever seqüências bastante complexas e em diferentes níveis de abstração. A representação temporal é descoberta utilizando a previsão como objetivo de aprendizado. A previsão também foi utilizada em (ELMAN, 1990) como heurística de geração de representações de seqüências temporais e o erro de previsão sugerido como um bom indicador da estrutura temporal existente nos dados. É portanto um método promissor para a resolução do problema apontado na introdução deste trabalho, que diz respeito à descoberta dos limites dos padrões temporais. Infelizmente, no entanto, não foram encontradas arquiteturas suficientemente capazes de resolver problemas reais da robótica, pois todos utilizam aprendizado em lote, aplicados em fases distintas para cada nível da arquitetura.

- As representações auto-organizáveis são bastante interessantes pelo fato de reduzirem a dimensionalidade dos dados de entrada e por serem geradas em poucas épocas de treinamento. Constatou-se que a maioria das arquiteturas gerou representações que distinguem as ambigüidades espaciais das seqüências temporais. No entanto, todos os modelos tiveram de ser testados com diversos parâmetros até que fossem encontrados valores que gerassem representações estáveis. Em nenhum dos artigos originais foram dadas heurísticas para a escolha adequada de parâmetros. Também não foi encontrado nenhum tipo de análise sobre os efeitos dos parâmetros na geração das representações temporais.

- As representações predeterminadas das memórias de curto prazo são instantaneamente geradas e exigem em geral, pouco processamento. Em compensação, seus parâmetros devem ser determinados com cuidado pelo projetista da rede,

para que representem as características temporais importantes do sinal de entrada. A escolha de parâmetros é realizada com base na experiência empírica do projetista ou com testes preliminares sobre os dados de entrada.

A escolha por representações temporais adequadas é uma tarefa complicada e deve ser feita com cuidado, pois existem vantagens e desvantagens em cada uma delas. Provavelmente as soluções mais robustas sejam obtidas pela utilização de mais de um tipo de representação temporal.

O que se percebeu com a análise dos modelos de memórias de curto prazo e mapas auto-organizáveis foi que a maioria deles utiliza a operação de integração temporal de forma direta ou indireta para a geração de representação temporal. A integração é útil para eliminação de ruído, pois representa o passado sob a forma de uma média. Em compensação, este tipo de informação não indica as mudanças ocorridas no passado. O modelo de neurônio proposto no próximo capítulo representa as mudanças do sinal e é baseado na operação de diferenciação.

3 NEURÔNIO DIFERENCIADOR-INTEGRADOR

A escolha da representação dos dados tanto de entrada como de saída é fundamental para o sucesso no treinamento de sistemas de aprendizagem, como as redes neurais. As redes neurais são treinadas para encontrar um mapeamento (possivelmente não linear) entre as entradas e as saídas. Este mapeamento deve também relacionar corretamente entradas nunca vistas, com saídas desejadas. No entanto, em muitos casos, os dados disponíveis não são suficientes para a criação deste mapeamento. Ou ainda, os dados disponíveis não se encontram na forma ideal para encontrá-lo. O modelo neural sugerido neste trabalho tem o objetivo de resolver o segundo caso, da mesma forma que as memórias de curto prazo, apresentadas no capítulo 2.

Antes de apresentar o modelo propriamente dito, será feita uma introdução ao assunto com o objetivo de clarear a evolução do próprio modelo até sua forma final.

3.1 Origem do Modelo

A idéia do modelo surgiu ao se tentar aplicar o princípio de Compressão de História, abordado na seção 2.3 no nível de neurônios artificiais. Supôs-se que o neurônio tenha o papel de um previsor, neste caso, toda a informação do sinal de entrada prevista não deve ser repassada para as camadas superiores da rede neural. Deve ser propagado somente o erro de previsão, ou seja, a novidade.

Como o objetivo da pesquisa sempre foi voltado às aplicações em robótica, os sinais de entrada imaginados foram os sinais sensoriais de robôs. Estes sinais temporais, por refletirem a realidade física, possuem propriedades muito semelhantes aos sinais sensoriais que os seres inteligentes processam. Portanto, supôs-se que o tipo de previsão realizada pelos neurônios artificiais deveria ser o mesmo existente nos neurônios sensoriais naturais.

Foram encontrados artigos de neuro-biologia com estudos estatísticos em fluxos sensoriais artificiais (seqüências de vídeo) confirmando a presença de forte correlação espaço-temporal nestes dados (DONG; ATICK, 1995a). Ou seja, as mudanças do fluxo sensorial tanto no espaço como no tempo são graduais e pequenas se observadas em uma curta janela de tempo. A correlação espaço-temporal indica existência de redundância neste tipo de sinal temporal.

A partir desta constatação, pode-se afirmar que, se os neurônios tiverem alguma forma de se adaptar ao estado atual do fluxo sensorial e responder somente às suas variações, estarão retirando do fluxo sensorial toda a redundância e correlação espaço-temporal, reduzindo ao máximo a informação necessária para representar o ambiente (DONG; ATICK, 1995b).

A correlação temporal pode ser eliminada de um sinal com a obtenção de sua derivada ou uma aproximação da derivada. Esta foi a fonte de inspiração para o modelo.

3.2 Diferenciação

Para extrair a informação de mudança de um sinal geralmente se aplica a *diferenciação de dois pontos adjacentes*, ou simplesmente, *diferenciação*. Este método consiste em discretizar o tempo em pequenos intervalos e computar a diferença (subtração) do valor de cada instante pelo valor do instante anterior. Esta operação pode ser feita de forma recursiva, ou seja, é possível diferenciar o sinal mais de uma vez e obter informações de segunda ordem ou ordens superiores.

A diferenciação é bastante utilizada para análise de séries temporais de baixa dimensionalidade, como valores da bolsa de mercado. No entanto, este método simples seria pouco eficiente se utilizado para filtrar dados antes de serem submetidos, por exemplo, ao processamento de uma rede neural. Existem dois motivos principais para isso:

- Se o sinal temporal apresentar autocorrelação significativa e por longo período de tempo (Ex.: uma senoidal com período de mil amostras como ilustrado na figura 3.1-A), terá como diferenciação, sinais de baixa amplitude (figura 3.1-B). Se a diferenciação for aplicada mais de uma vez, pode resultar em valores muito próximos de zero (figura 3.1-C). Esta informação será praticamente menosprezada pela rede neural, frente aos valores dos outros sinais de entrada. Seria necessário diferenciar toda a seqüência temporal para então normalizá-la, pois, como é sabido, para se obter um treinamento eficiente de redes neurais, aconselha-se utilizar dados normalizados (HAYKIN, 1999). Esta solução não poderia ser utilizada em caso de aprendizado incremental, já que o conjunto de treinamento não necessariamente é conhecido *a priori*.

- Se o sinal temporal for ruidoso (figura 3.1-D), a sua diferenciação resultará também em um sinal ruidoso, muito provavelmente com perda de informação, pois a magnitude do ruído será maior que a magnitude da diferenciação do sinal sem o ruído (figuras 3.1-E e 3.1-F).

Torna-se evidente a necessidade de outro tipo de aproximação para a computação de derivada, que seja tolerante a ruídos e não gere perda de amplitude no sinal.

3.3 Aplicação do Modelo

Antes de apresentar o modelo, é importante ressaltar o papel do modelo na resolução de problemas e a forma como ele pode ser utilizado em conjunto com as arquiteturas já existentes da Inteligência Computacional.

O modelo neural sugerido neste trabalho, denominado Neurônio Diferenciador-Integrador (NDI), foi desenvolvido como um componente básico que pode ser incluído em arquiteturas conexionistas clássicas para melhorar o espaço de representação dos dados de entrada, facilitando o aprendizado. Sua função básica é extrair informações temporais implícitas do sinal de entrada original.

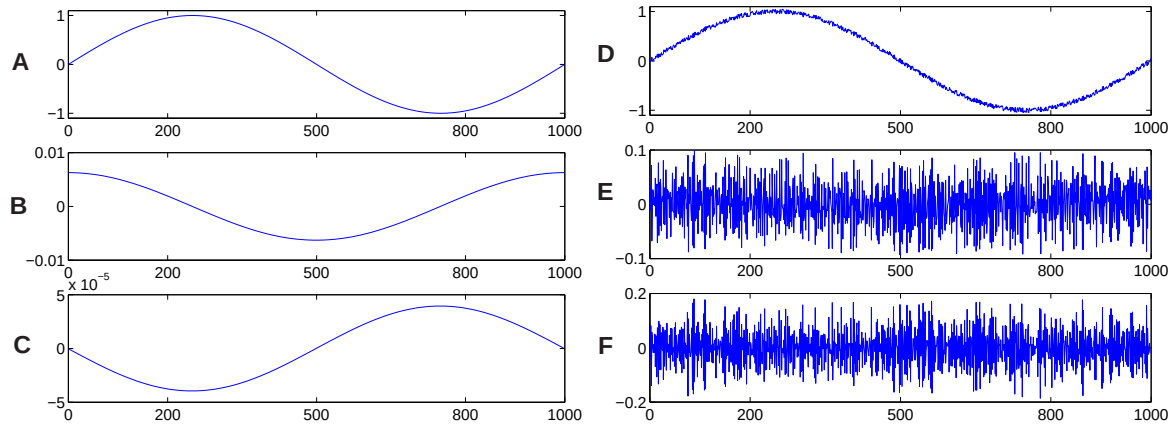

Figura 3.1: (A) Sinal senoidal de período mil. (B) Diferenciação do sinal senoidal. (C) Diferenciação segunda do sinal senoidal. (D) Mesmo sinal somado de ruído branco de valores [-0,05; 0,05]. (E) Diferenciação do sinal senoidal ruidoso. (F) Diferenciação segunda do sinal senoidal ruidoso

A figura 3.2 ilustra o uso do modelo neural em um *Perceptron de Múltiplas Camadas*. Os Neurônios Diferenciadores-Integradores recebem como entrada, a informação dos neurônios sensoriais e aumentam o espaço de entrada para os neurônios computacionais da rede. Da mesma forma, os NDIs poderiam ser aplicados a mapas auto-organizáveis ou sistemas de extração de modelos de agrupamentos como o ARAVQ, *Fuzzy ART* ou alguma outra variação, como apresentado no capítulo 4.

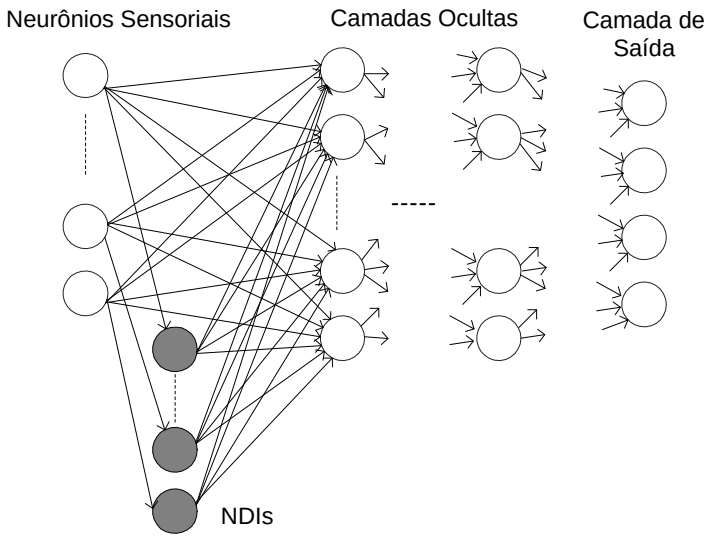

Figura 3.2: Neurônios NDI aplicados em um *perceptron de múltiplas camadas*.

O modelo foi desenvolvido visando a aplicação em problemas que exijam aprendizagem incremental, como em processamento sensorial de robôs ou qualquer processamento de dados em tempo real. Portanto, primou-se por um modelo de baixa complexidade computacional e de resposta instantânea.

3.4 Definição do Modelo

Para resolver as duas limitações citadas na seção 3.2, desenvolveu-se um modelo neural genérico e parametrizável. Para sua implementação é necessário duas memórias e seu cálculo é simples e incremental. A figura 3.3 apresenta o modelo na forma de diagrama. As equações envolvidas seguem abaixo:

$$\overline{x}(t) = (1-\mu)x(t) + \mu\overline{x}(t-1) \tag{3.1}$$

$$dx(t) = x(t) - \overline{x}(t-1) \tag{3.2}$$

$$\overline{dx}(t) = (1-\eta)dx(t) + \eta\overline{dx}(t-1) \tag{3.3}$$

Onde, $x(t)$ é a entrada do neurônio e $\overline{dx}(t)$ é a sua saída. μ e η são constantes entre 0 e 1. Considera-se $\overline{x}(0) = 0$ e $\overline{dx}(0) = 0$.

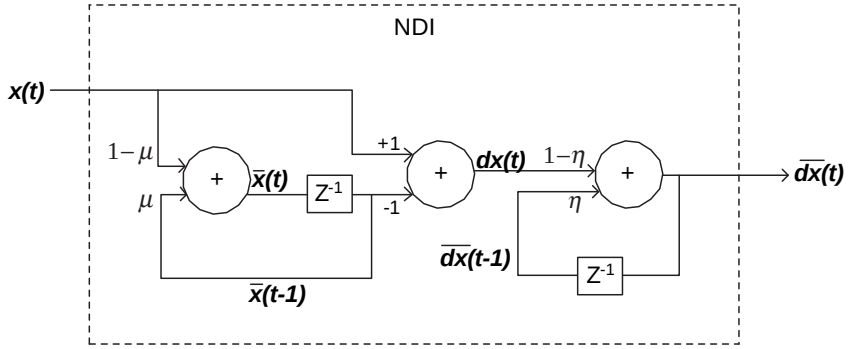

Figura 3.3: Modelo do Neurônio Diferenciador-Integrador (NDI)

As equações 3.1 e 3.2 do Neurônio Diferenciador-Integrador refletem as seguintes idéias: Como o sinal temporal apresentado na seção 3.2 varia muito pouco entre cada amostra, sua diferenciação resulta em um sinal de pequena amplitude. Se, ao invés de se fazer a diferença entre dois instantes de tempo consecutivos, for feita a diferença entre o sinal no tempo atual e o sinal de T instantes anteriores (onde $T > 1$), o resultado será um sinal de maior amplitude que a diferenciação e refletirá a variação do sinal numa janela de tempo maior que um. No entanto, para implementar este procedimento, seria necessário manter em memória os últimos$T$ dados de entrada.

Uma alternativa para a idéia anterior seria computar a diferença entre o sinal do tempo atual pela média dos T sinais mais recentes. O sinal resultante seria menos sensível a ruído que o obtido pela primeira alternativa, mas também seria necessário manter em memória os últimos T dados de entrada. Ao invés disso, poder-se-ia utilizar a média móvel do sinal. A média móvel é computada como a integração do sinal no tempo, utilizando coeficientes exponenciais que tendem a zero para os dados mais remotos. Seu cálculo é simples e incremental e exige somente uma unidade de memória. As Memórias Exponenciais (abordadas na seção 2.1.2) realizam exatamente o mesmo tipo de cálculo. O cálculo de $\overline{x}(t)$ na equação 3.1 representa a média móvel do sinal com a constante de decaimentoμ e a computação de $dx(t)$ na equação 3.2 é a diferença do sinal por sua média móvel.

A diferença do sinal por sua média móvel resolve o primeiro problema apontado na seção 3.2 relativo à pequena amplitude da diferenciação. No entanto, se o sinal for, ruidoso, como no segundo caso apontado, o ruído permanecerá na saída. Para

que o ruído seja reduzido, é necessário aplicar outra média móvel, desta vez, ao sinal $dx(t)$, resultando na equação 3.3 para $\overline{dx}(t)$.

A figura 3.4 apresenta o resultado da aplicação do sinal senoidal de período mil, com e sem ruído, a dois Neurônios Diferenciador-Integrador encadeados, ambos com os parâmetros $\mu = 0{,}98$ e $\eta = 0{,}8$. Ao comparar seus resultados com a primeira e segunda diferenciação ilustradas na figura 3.1, pode-se concluir que:

- A saída do neurônio tem amplitudes menores que o sinal de entrada, mas a perda não é tão grande quanto a diferenciação tradicional.
- Ao contrário da diferenciação, o ruído existente no sinal original foi reduzido na saída do neurônio.
- A resposta do neurônio está atrasada. Idealmente o ponto mínimo da saída do primeiro neurônio deveria ser no instante $t = 500$, como na figura 3.1-B, mas este apresenta-se aproximadamente no instante t = 550 (figura 3.4-B). Este atraso corresponde a $\frac{1}{20}$ do período do sinal original, e pode ser tolerável na maioria dos problemas, como foi constatado nos experimentos.
- Nos primeiros 100 instantes de tempo o sinal de saída é instável e não corresponde a uma informação de mudança confiável. Isso se dá porque as médias móveis dos integradores $\overline{x}$ e $\overline{dx}$ ainda não coletaram informação suficiente do fluxo de entrada. Esta perturbação só ocorre no início da apresentação da seqüência temporal.

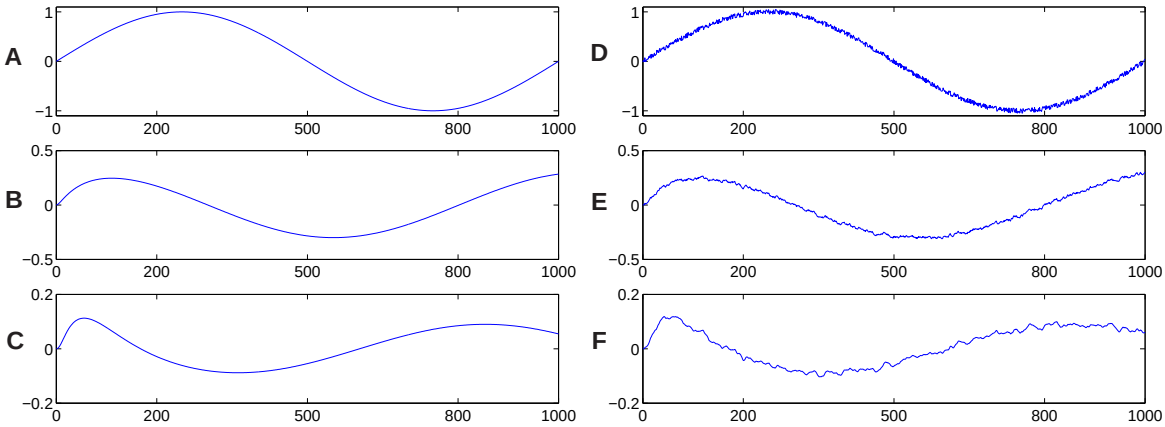

Figura 3.4: (A) Sinal senoidal de período mil. (B) Saída do primeiro NDI pela aplicação do sinal senoidal. (C) Saída do segundo NDI (recebe como entrada a saída do primeiro NDI) (D) Mesmo sinal somado de ruído branco de valores [-0,05; 0,05]. (E) Saída do primeiro NDI pela aplicação do sinal senoidal ruidoso. (F) Saída do segundo NDI (recebe como entrada a saída do primeiro NDI)

Para evidenciar o comportamento de diferenciação do modelo neural, aplicou-se diferentes sinais periódicos que são normalmente usados em análise de circuitos temporais, como os RC. As respostas obtidas pelo NDI de parâmetros $\mu = 0{,}6$ e $\eta = 0{,}4$ a estes sinais podem ser observadas na figura 3.5. A saída observada é muito semelhante ao comportamento de diferenciadores RC. Estes componentes digitais são utilizados para realizar a diferenciacão de sinais elétricos e por serem

constituídos de materiais físicos, apresentam capacitância e resposta não linear, razão pela qual amenizam as mudanças bruscas do sinal de entrada. O paralelo entre o NDI e os diferenciadores RC será revisto na próxima seção.

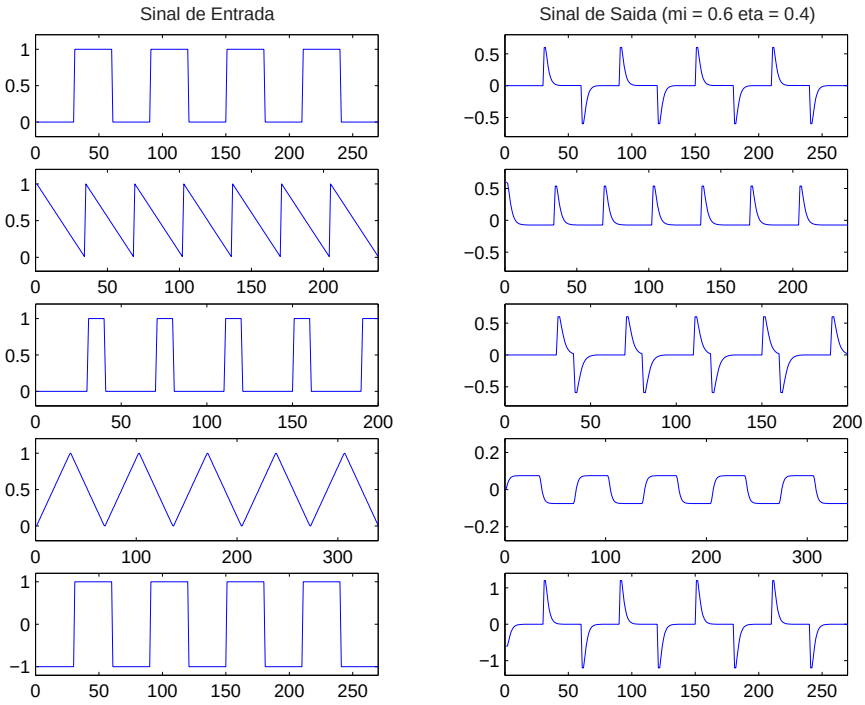

Figura 3.5: Diferentes sinais aplicados a um NDI de parâmetros $\mu = 0{,}6$ e $\eta = 0{,}4$.

3.5 Propriedades do Modelo

O modelo neural proposto pode ser analisado sob a forma de integradores e diferenciadores de circuitos eletrônicos. Separadamente, ambos já foram sugeridos como componentes fundamentais para a criação de sistemas neurais em circuitos VLSI analógicos (do inglês, *very large scale integration*)(MEAD, 1989).

O Neurônio Diferenciador-Integrador se comporta como um diferenciador e/ou um integrador, dependendo dos parâmetros η e μ escolhidos.

- Para $\eta = 0$ e $\mu < 1$, seu comportamento é de um diferenciador RC. As equações do NDI são reduzidas a uma subtração do sinal por sua integração.
- Para $\eta < 1$ e $\mu = 1$, seu comportamento é o de um integrador RC. As equações do NDI são reduzidas a uma integração do sinal de entrada.
- Para η e μ entre 0 e 1, o neurônio funciona como um diferenciador e um integrador.
- No caso especial $\eta = 0$ e $\mu = 0$, as integrações de $\overline{x}(t)$ e $\overline{dx}(t)$ são anuladas resultando na equação $x(t) - x(t-1)$ que é a *diferenciação de dois pontos adjacentes*.

Para caracterizar melhor o funcionamento deste neurônio, pode-se rescrever a função $\overline{dx}$ como a convolução do sinal de entrada $x(t)$ usando a função *kernel* $c(t)$ (equação 3.4).

$$\overline{dx}(t) = \sum_{\tau=1}^{t} c(t-\tau)x(\tau) \tag{3.4}$$

A função *kernel* determina a participação dos sinais anteriores na computação da saída do NDI no instante atual. A partir de operações aritméticas sobre as equações do modelo NDI, se obteve a função *kernel* do NDI (equação 3.5).

$$c(t) = (1-\eta)\eta^t - (1-\eta)(1-\mu)\sum_{j=0}^{t-1}(\eta^{t-1-j}\mu^j) \tag{3.5}$$

A figura 3.6 representa graficamente a função *kernel* $c(t)$ para alguns valores de η e μ. É importante ressaltar que o parâmetro t da função se refere a tempo decorrido. A função *kernel* representa a *função de transferência* do NDI, que em Técnicas Digitais, é definida como a resposta a um impulso unitário do sistema.

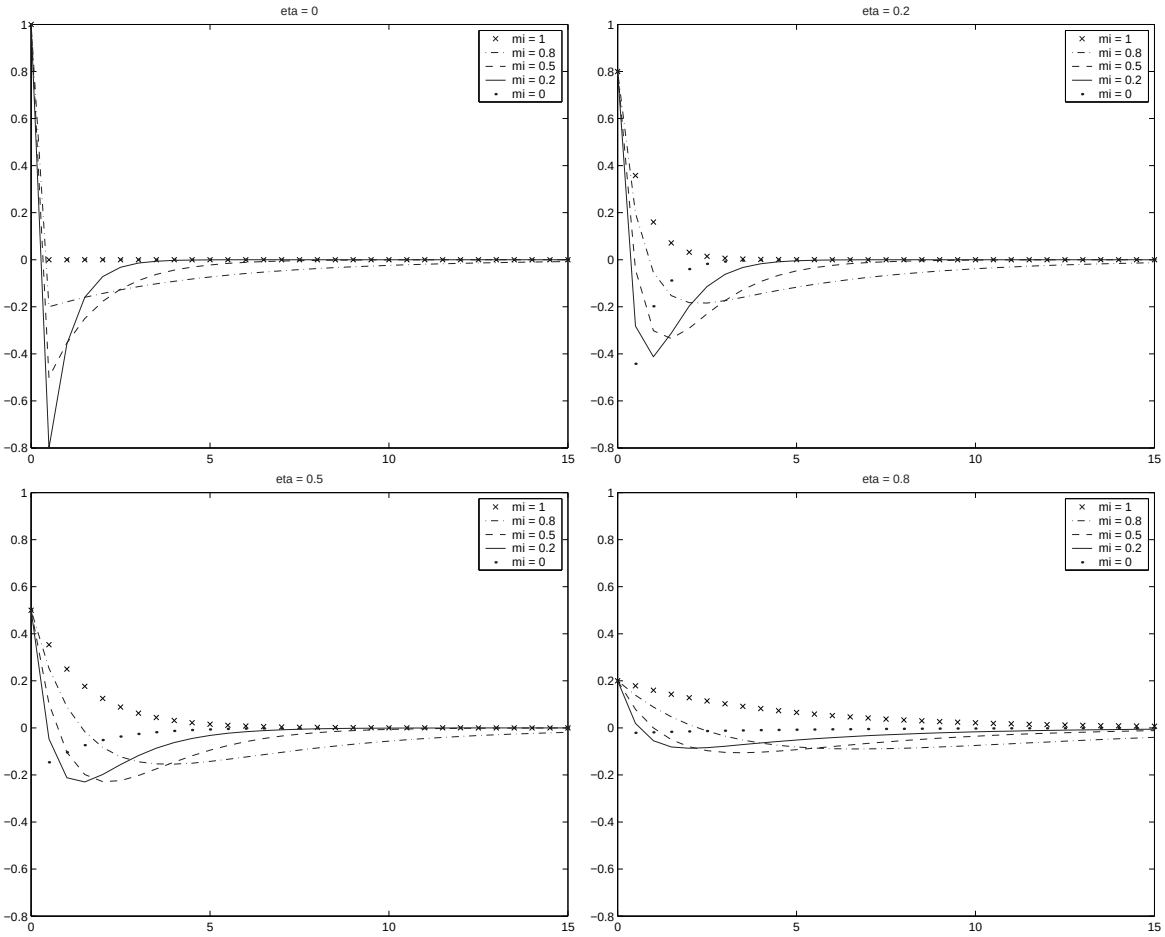

Figura 3.6: Função *kernel* do NDI com $\eta = 0{,}0; 0{,}2; 0{,}5$ e $0{,}8$.

A função *kernel* do NDI tem um comportamento padrão independente dos parâmetros escolhidos. Excetuando os casos em que $\mu = 1$, os sinais mais recentes possuem peso positivo e os sinais que os antecedem, possuem peso negativo significativo. O restante dos sinais possui peso negativo próximo de zero. Ou seja, a convolução do sinal utilizando a função *kernel* computa a diferença dos sinais mais recentes pelos sinais que os antecedem, reproduzindo, de forma mais genérica, o cálculo da *diferenciação*. Para os casos em que $\mu = 1$, a função *kernel* é sempre

positiva e tende a zero em uma taxa exponencial, refletindo o comportamento de integração já apontado.

Outra observação importante a respeito da função *kernel* é relativa à sua área total. Excetuando o caso $\mu = 1$, a integral da função *kernel* é sempre zero, ou seja, a região positiva tem a mesma área que a região negativa. Isto faz com que, se um sinal constante for aplicado ao NDI, a saída do neurônio tenderá a zero. Este comportamento corresponde exatamente ao esperado, pois um sinal constante é totalmente previsível.

3.5.1 Amplitude do Sinal

A partir da análise da função *kernel* é possível determinar a amplitude máxima do sinal de saída do NDI em relação à amplitude do sinal de entrada. A determinação da amplitude é importante pois deve ser adequada às limitações da arquitetura conexionista em vista. Em geral, os dados de entrada para uma rede neural devem ser normalizados em intervalos de 0 a 1 ou -1 a 1. O NDI deve portanto respeitar tais limites de representação.

Para os casos em que o NDI funciona como um diferenciador ($\mu < 1$), a sua função *kernel* é positiva para o parâmetro $t = 0$ e torna-se negativa a partir de um instante k, onde $k > 0$. O valor de k depende dos parâmetros η e μ do NDI. Os k dados de entrada mais recentes terão pesos positivos no cálculo da saída do NDI e o restante terá peso negativo. Logo, para determinar o valor máximo na saída de um NDI no tempo t, cujo intervalo de entrada seja $[\alpha,\beta]$, basta aplicar ao NDI um sinal em forma de onda quadrada, onde o valor inicial seja α e a partir do instante $t - k + 1$ seja igual a β.

Para ilustrar a conclusão anterior, escolheu-se como exemplo um NDI com $\eta =$ 0,5 $\mu =$ 0,75. A função *kernel* deste neurônio é apresentada na figura 3.7-A. As barras verticais da figura correspondem aos valores da função *kernel* nos instantes de tempo discretizados. Pode-se perceber, que a função torna-se negativa a partir do instante $t = k = 2$, ou seja, o sinal de entrada dos últimos dois instantes de tempo são multiplicados por pesos positivos enquanto que todos os outros são multiplicados por pesos negativos. Supondo que o sinal de entrada para este neurônio esteja compreendido no intervalo $[-1; 1]$ ($\alpha = -1$ e $\beta = 1$), o sinal que maximiza a saída deve ter o valor máximo 1 nos últimos dois instantes e o valor mínimo -1 nos demais, tal como ilustrado na figura 3.7-B. A saída do NDI pela aplicação deste sinal é apresentada na figura 3.7-C. O valor de saída do NDI após a apresentação de todo o sinal de entrada (instante $t = 0$) ultrapassou o limite superior de representação do sinal de entrada, resultando em 1,25.

Generalizando, para descobrir o intervalo de saída de um NDI ($[\alpha_{NDI},\beta_{NDI}]$), basta utilizar as equações 3.7 e 3.8, onde p corresponde à região positiva da função *kernel* discretizada no tempo e é calculada com a equação 3.6. Estas equações utilizam somente a região positiva porque, como já foi afirmado, é igual à região negativa.

$$p = \sum_{\tau=0}^{k-1} c(\tau) \tag{3.6}$$

$$\alpha_{NDI} = (\alpha - \beta) * p \tag{3.7}$$

$$\beta_{NDI} = (\beta - \alpha) * p \tag{3.8}$$

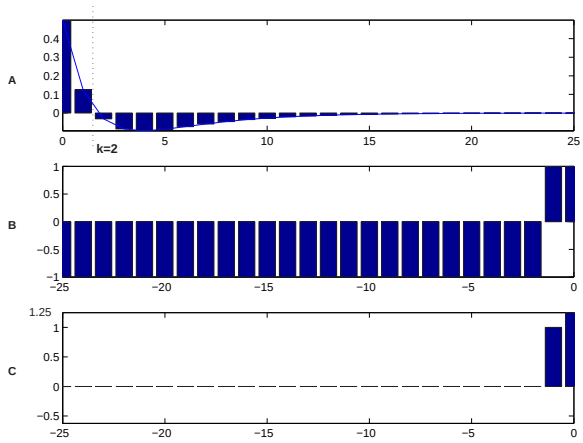

Figura 3.7: (A) Função *kernel* relativa ao tempo atual t do NDI com $\eta = 0{,}5$ $\mu = 0{,}75$. (B)Sinal de entrada aplicado ao NDI. (C) Saída do NDI.

Voltando ao caso anterior, o NDI da figura 3.7-A possui $p = 0{,}625$, correspondente a soma da função *kernel* para $t = 0$ e $t = 1$. O sinal de entrada aplicado estava contido no intervalo $[-1{,}1]$, o que resulta em $\beta_{NDI} = 2 * 0{,}625 = 1{,}25$.

A região positiva da função *kernel* varia conforme os parâmetros η e μ escolhidos e possui uma relação não linear com estes parâmetros. Sua superfície é ilustrada na figura 3.8. Baseando-se nesta superfície e no intervalo dos dados de entrada é possível descobrir o intervalo dos dados de saída do NDI e portanto, normalizá-lo em tempo real com uma constante multiplicativa associada à saída do NDI.

A normalização por multiplicação não foi utilizada em todos os experimentos, somente nos casos em que a área positiva p ficou próxima de zero o que resultaria em sinais de pequena amplitude. Nestes casos, utilizou-se constantes maiores que um para ampliar o sinal, e garantir amplitude próxima dos valores originais.

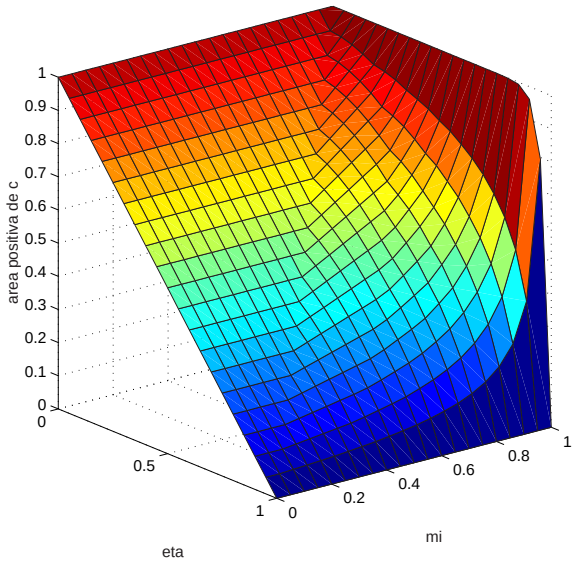

Figura 3.8: Região positiva da função *kernel* em relação aos parâmetros μ e η do NDI.

3.6 Plausibilidade Biológica

O modelo do Neurônio Diferenciador-Integrador pode ser considerado biologicamente plausível se for levado em conta a possível influência dos diferentes neurotransmissores existentes no cérebro sobre a atividade de um neurônio. Sabe-se que cada neurotransmissor pode desencadear reações de curta ou longa duração, ou seja, a informação computada num neurônio não se resume aos estímulos elétricos de cada instante de tempo, mas também depende dos processos químicos que estão ocorrendo no interior do neurônio e que carregam informação do passado. Fazendo um paralelo com a equação do modelo, bastariam dois tipos de neurotransmissores para implementar o neurônio. O efeito dos dois neurotransmissores sobre a célula é análogo às duas integrações temporais sugeridas no NDI. A integração é um processamento tipicamente encontrado em modelos neurais conhecidos como *Leaky Integrators* (ARBIB, 1995).

Em um artigo recente (TRUCCOLO; DONG, 2001), foi sugerido que a região do cérebro chamada LGN (do inglês, *Lateral Geniculate Nucleous*) conhecida como a interface entre a retina e o córtex visual, tenha um papel importante na geração de uma representação ótima (*optimal coding*) da informação sensorial no tempo, pelo ponto de vista da Teoria da Informação. Como o sinal visual enviado à LGN é redundante no tempo e pouco eficiente, foi sugerido que a LGN realiza decorrelação temporal sobre o mesmo. O artigo investiga a possibilidade de realização da decorrelação pelo uso dos canais iônicos da célula, mais particularmente, os canais de Cálcio.

O *kernel* proposto pelos autores do artigo para obter a decorrelação temporal possui uma forma muito semelhante ao *kernel* do modelo sugerido no presente trabalho (ver figura 3.9), o que faz muito sentido, uma vez que a decorrelação temporal implica em eliminação da média do sinal, justamente o que é feito nas equações do NDI, mas com o objetivo de obter a informação de mudança.

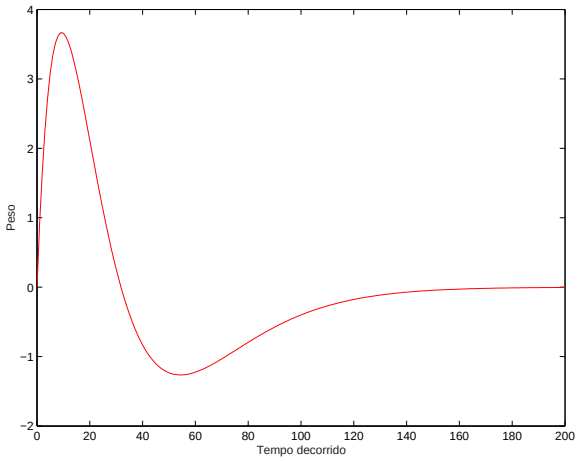

Figura 3.9: Função *kernel* adotada por Dong e seus colaboradores para obter decorrelação temporal.

Sob o ponto de vista de plausibilidade biológica, as operações realizadas pelo NDI mais provavelmente ocorrem nas sinapses, como ilustrado na figura 3.10. No entanto, a implementação deste modelo não seria tão facilmente adaptável a arquiteturas conexionistas existentes. Além disso, demandaria mais processamento,

pois cada neurônio teria de computar independentemente seus sinais de entrada. As duas dificuldades são resolvidas se o processamento da diferenciação for isolado em unidades específicas, como foi sugerido neste capítulo.

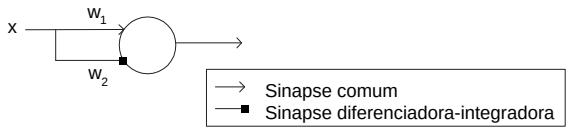

Figura 3.10: Modelo de sinapses que computam a diferenciação como o NDI.

3.7 Filtros Digitais e o NDI

Existe um vasto campo de estudo voltado para filtros digitais. São ferramentas importantíssimas para processamento de sinais. O estudo de filtros envolve a caracterização da sensibilidade a freqüências básicas. Esta abordagem é muito diferente das utilizadas na Inteligência Computacional, pois as propriedades dos sinais e dos filtros são analisadas principalmente no domínio freqüência e não no domínio tempo.

Todo o sinal temporal pode ser decomposto em séries periódicas (senos e cossenos) de diferentes freqüências, utilizando a *Transformada de Fourier*. A caracterização dos filtros é baseada na resposta a estas freqüências básicas. Basicamente existem três tipos de filtros: filtros passa-baixa, passa-alta e passa-faixa. Os filtros passa-baixa reproduzem na saída somente as freqüências do sinal de entrada que estão abaixo da freqüência de *corte* especificada. Os filtros passa-alta fazem exatamente o oposto, reproduzem somente as freqüências superiores à freqüência de *corte*. Os filtros passa-faixa, reproduzem um intervalo de freqüências e geralmente são construídos a partir de dois filtros mais simples, um passa alta e um passa baixa.

O neurônio Integrador-Diferenciador pode ser interpretado como um filtro digital. Dependendo dos parâmetros, o tipo do filtro muda. Para compreender melhor as características do NDI sob o ponto de vista de filtros digitais, são apresentadas na figura 3.11, o *módulo das respostas em freqüência* do filtro, obtidas a partir da aplicação da *transformada de Fourier* sobre a função *kernel* do modelo, variando-se os parâmetros μ e η exatamente como na figura 3.6.

Os gráficos da figura 3.11 apresentam no eixo horizontal a freqüência normalizada, independente da freqüência de amostragem. Isto é, considera-se que o tempo é discretizado e o sinal coletado em amostras a cada intervalo de tempo, portanto a maior freqüência reconhecível é de 0,5, correspondendo a um sinal com período de duas amostras (por exemplo, $-1, 1, -1, 1$,...). O eixo vertical corresponde a amplitude da saída de cada freqüência básica em relação à sua amplitude no sinal original.

A análise do *módulo da resposta em freqüência* do NDI revela propriedades diferentes conforme os parâmetros de μ e η:

- Para $\eta = 0$ e $\mu < 1$, seu comportamento é de um filtro passa-alta. As freqüências mais baixas têm sua amplitude fortemente reduzida.

- Para $\eta < 1$ e $\mu = 1$, seu comportamento é o de um filtro passa-baixa. As freqüências mais altas sofrem redução de amplitude.

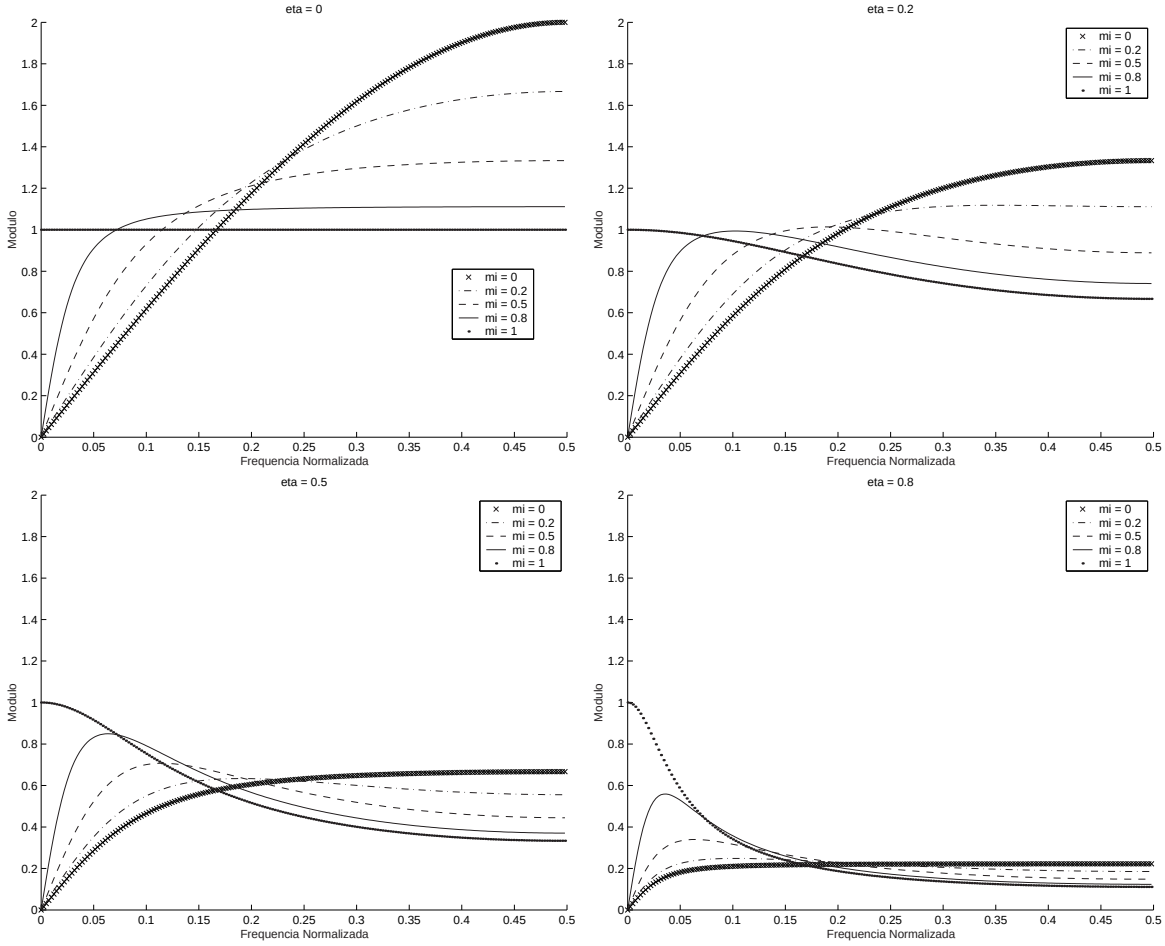

Figura 3.11: *Módulo da resposta em freqüência* do NDI com $\eta = 0{,}0; 0{,}2; 0{,}5$ e 0,8.

- Para η e μ entre 0 e 1, o neurônio funciona como um filtro passa-faixa. Em uma determinada freqüência, o neurônio atinge a maior resposta e a partir deste ponto há um declínio gradual.

Facilmente pode-se fazer um paralelo entre filtros digitais e os componentes básicos de Técnicas Digitais. Os Integradores RC são utilizados para implementar filtros passa baixa, enquanto que os Diferenciadores RC são utilizados para implementar filtros passa alta.

Além do *módulo*, também se analisa a *fase da resposta em freqüência*, ou seja, a alteração de fase das freqüências básicas do sinal de entrada. A figura 3.12 apresenta esta informação para as mesmas variações dos parâmetros μ e η.

Para clarear estes conceitos será analisado o caso especial da *diferenciação simples* comentado na seção 3.2 e simulado pelo NDI com os parâmetros $\mu = 0$ e $\eta = 0$. O sinal senoidal de período 1000, apresentado na figura 3.1-A, é um exemplo de sinal com somente uma freqüência básica, simples de analisar. Sua freqüência é de 0,001.

Pela figura 3.12, a mudança de fase para o caso $\mu = 0$ e $\eta = 0$ de um sinal com freqüência 0,001 é de 90 graus. Como pode-se constatar pela figura 3.1-B, o resultado obtido pela diferenciação do sinal corresponde ao sinal original defasado de 90 graus. A redução da amplitude do sinal pode ser explicada pelo *módulo da resposta em freqüência* apresentado na figura 3.11. A freqüência 0,001 possui

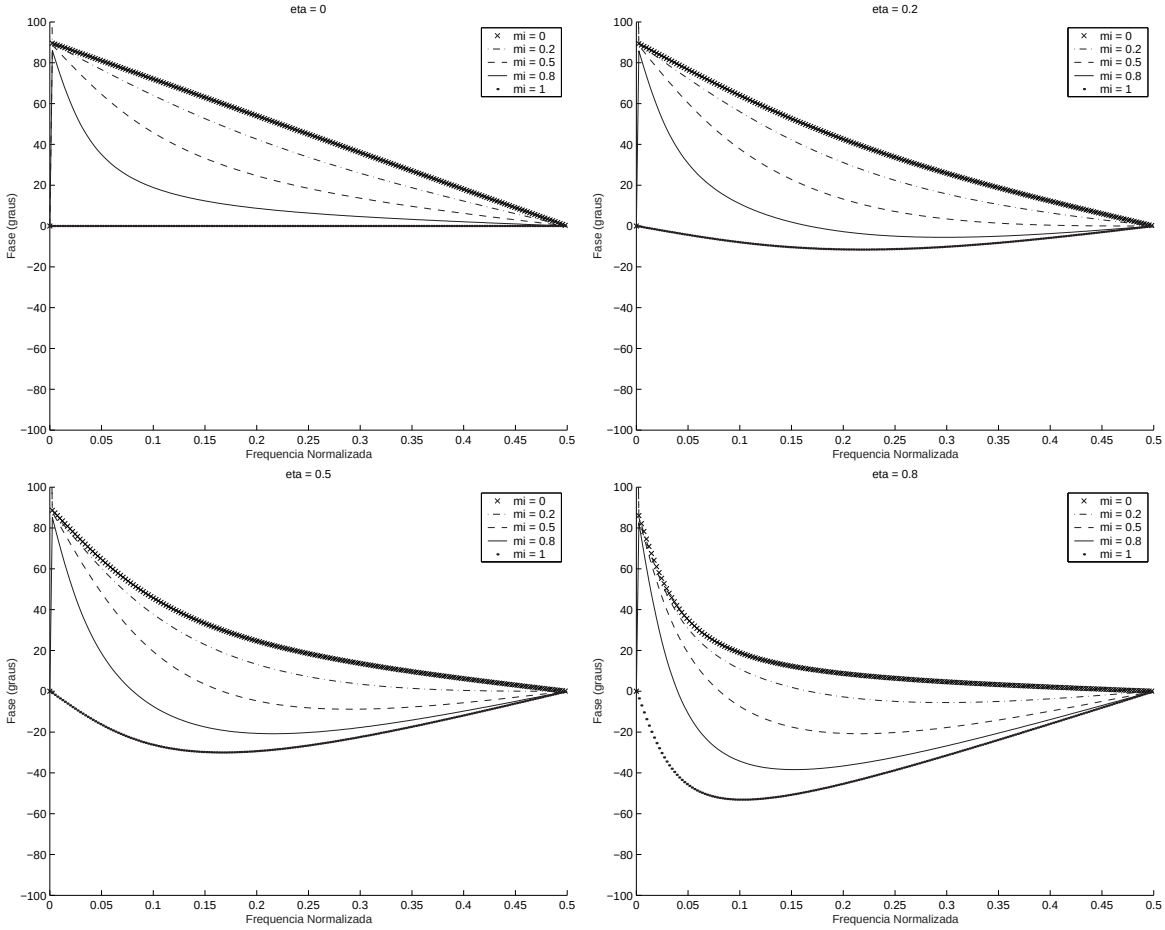

Figura 3.12: *Fase da resposta em freqüência* do NDI com $\eta = 0{,}0; 0{,}2; 0{,}5$ e $0{,}8$.

resposta de aproximadamente 0,06, exatamente a amplitude do sinal de saída que se obteve na figura 3.1-B.

Para se compreender a alta sensibilidade a ruídos retratadas na figura 3.1-E, basta examinar novamente a figura 3.11 para o caso das altas freqüências. A *diferenciação* amplifica as altas freqüências, no máximo, duplicando sua amplitude.

Ainda analisando o caso da *diferenciação*, pela figura 3.12, verifica-se que a mudança de fase não é igual para todas as freqüências. Ela começa em 90 graus e decai linearmente a zero na freqüência máxima de 0,5. Isso significa que a diferenciação só é uma boa aproximação para a derivada do sinal se este não possuir altas freqüências. Na maioria dos sinais estudados, as freqüências no início do intervalo são as mais importantes e portanto, a *diferenciação* pode ser utilizada.

Para ilustrar a utilidade da análise do NDI sob o ponto de vista de filtros digitais, gerou-se um sinal composto por uma senoidal de baixa freqüência somada com outra de alta freqüência, e a partir das respostas do NDI das figuras 3.11 e 3.12, escolheu-se parâmetros adequados para computar uma aproximação da derivada do sinal de baixa freqüência com eliminação do sinal de alta freqüência, considerado ruído.

Claramente a escolha de parâmetros para o NDI deve levar em conta dois objetivos conflitantes: a redução de ruído, observada pela diferença da resposta entre altas e baixas freqüências da figura 3.11 e a capacidade de computar uma aproximação da derivada do sinal, observada pela mudança de fase das freqüências (figura 3.12, que idealmente deveria ser de 90 graus.

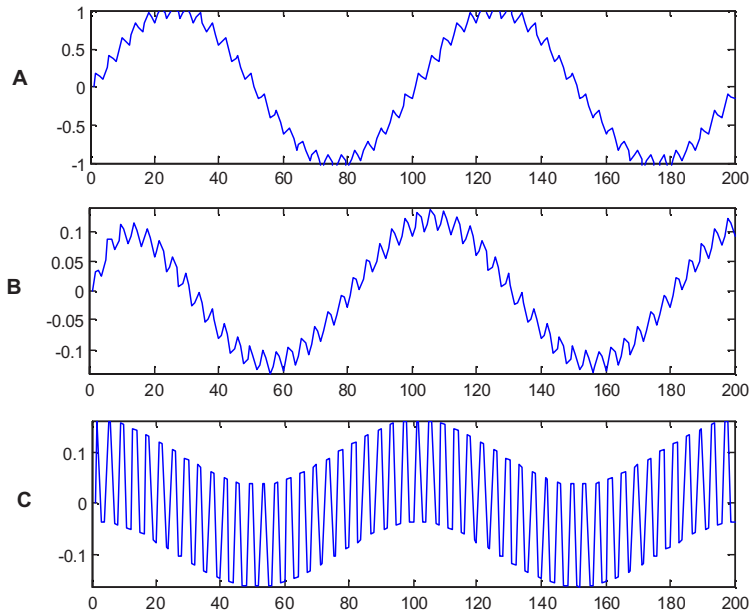

Figura 3.13: (A) Sinal de entrada composto por duas senoidais de freqüências0,01 e 0,25. (B) Saída do NDI com $\eta = 0{,}8$ e $\mu = 0{,}5$. (C) Resultado da diferenciação do sinal composto.

O sinal é ilustrado na figura 3.13-A. A senoidal de baixa freqüência tem período de 100 amostras, ou seja, freqüência 0,01 e a senoidal de alta freqüência tem período de 4 amostras com freqüência 0,25. Pelas figuras 3.11 e 3.12, escolheu-se os parâmetros $\eta = 0{,}8$ e $\mu = 0{,}5$ pois corresponde a um filtro que oferece uma boa relação entre mudança de fase e redução do ruído. A alta e a baixa freqüência perdem relativamente a mesma amplitude (mantendo o ruído em níveis toleráveis) e a mudança de fase da freqüência 0,01 é de aproximadamente 70 graus, que não é ótimo mas adequado. O resultado é apresentado na figura 3.13-B. É importante notar que os dois gráficos estão em escalas diferentes, e o ruído na saída apesar de aparentar, não sofre amplificação. A figura 3.13-C apresenta o resultado da *diferenciação* do sinal composto para fins de comparação. Trata-se de outro exemplo que evidencia a característica de ampliação de altas freqüências da *diferenciação*.

O modelo neural proposto apresenta grandes vantagens em relação à *diferenciação simples*, no entanto, a separação de freqüências não é muito evidente e a freqüência de *corte* não é fácil de se determinar a partir dos parâmetros de μ e η. O comportamento do NDI sob o ponto de vista de filtros digitais está longe do ótimo que pode ser alcançado com escolhas precisas dos parâmetros dos filtros digitais. Entretanto, o NDI representa um compromisso entre simplicidade e desempenho que é bastante adequado para as soluções conexionistas estudadas.

Os experimentos apresentados no próximo capítulo utilizam NDIs em conjunto com arquiteturas conexionistas para resolver problemas temporais. É provável que os mesmos resultados possam ser obtidos utilizando filtros digitais convencionais no lugar dos Neurônios Diferenciadores-Integradores. No entanto, seria preciso determinar dentre um conjunto muito grande de possibilidades, os parâmetros e os tipos de filtros utilizados em cada situação.

4 RESULTADOS EXPERIMENTAIS

O Neurônio Diferenciador-Integrador foi desenvolvido como um neurônio filtro, ou preprocessador de sinais. Sua função, como já foi mencionado, é de gerar informação sobre a variação do sinal, em uma janela de tempo particular, com tolerância a ruídos. Por ser de simples implementação e modular, este neurônio pode ser facilmente incorporado em qualquer arquitetura conexionista conhecida.

Neste capítulo o modelo sugerido será utilizado em diferentes arquiteturas conexionistas para gerar representações temporais úteis que acelerem ou melhorem a generalização da solução. Como o objetivo do trabalho é validar o modelo, foram escolhidos experimentos teóricos, relativamente simples que pudessem ser implementados e simulados em MATLAB utilizando arquiteturas simples e bem conhecidas. No entanto, acredita-se que problemas práticos de maior complexidade, principalmente da robótica, poderiam ser resolvidos pelas mesmas técnicas empregadas neste capítulo.

4.1 Previsão de Séries Temporais Caóticas

As séries temporais caóticas são largamente utilizadas e representam problemas de difícil tratamento pois, como é sabido, pequenos erros de medida causam grandes erros de previsão. A série caótica de Mackey-Glass é conhecida como um *benchmark* de sistemas de previsão. Para um resumo de resultados de previsão da série de Mackey-Glass em diversas arquiteturas conexionistas e não-conexionistas, consultar (GERS; ECK; SCHMIDHUBER, 2001).

4.1.1 Geração da Seqüência

A série de Mackey-Glass é definida pela equação diferencial 4.1. Os parâmetros geralmente utilizados são: $\alpha = 0{,}2$, $c = 10$, $\tau = 17$ e $\beta = 0{,}1$, com $x(0) = 1{,}2$. São geradas 2000 amostras da série temporal, sendo 500 amostras ($124 \leq t \leq 623$) extraídas para treinamento e outras 500 amostras ($624 \leq t \leq 1123$) para o teste de generalização.

$$\dot{x}(t) = \frac{\alpha x(t-\tau)}{1 + x^c(t-\tau)} - \beta x(t) \tag{4.1}$$

A figura 4.1 mostra o conjunto de amostras da série de Mackey-Glass geradas para treinamento e teste. Apesar de aparentemente regular, esta série temporal é caótica, pois existem pequenas diferenças entre cada ciclo da freqüência principal claramente reconhecível.

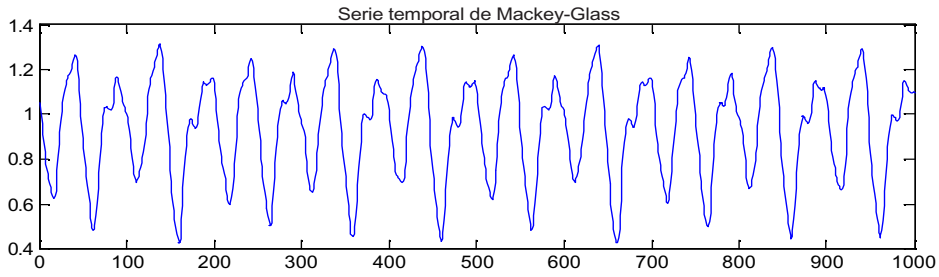

Figura 4.1: 1000 amostras da série temporal de Mackey-Glass para treinamento e teste.

Os experimentos de previsão encontrados na literatura científica para esta série consistem em prever a cada instante t, a entrada do instante $t+k$, onde geralmente k é 1, 6 ou 84. Neste trabalho será estudado somente o caso $k = 6$, pois seus resultados experimentais existem em maior quantidade.

4.1.2 Arquiteturas de Treinamento

Para este problema, foram implementados em MATLAB quatro sistemas de aprendizado, sendo que dois deles, utilizam NDIs.

O primeiro sistema foi escolhido porque é um exemplo típico de solução conexionista usada em problemas temporais. Trata-se de um *Perceptron de Múltiplas Camadas* com linha de atraso na entrada, como mostrado na figura 4.2. A camada oculta tem cinco neurônios e a linha de atraso é composta por três memórias que disponibilizam para a rede o sinal atrasado em 6, 12 e 18 instantes de tempo. Esta configuração da linha de atraso foi encontrada na maioria das arquiteturas utilizadas com esta série temporal.

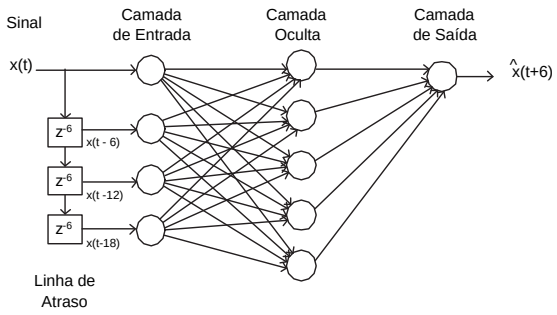

Figura 4.2: MLP com linhas de atraso para previsão da série de Mackey-Glass.

O segundo sistema utiliza uma arquitetura híbrida de lógica *Fuzzy* e redes neurais e é denominada ANFIS (do inglês *Adaptive-Network-based Fuzzy Inference System*) (JANG, 1993). Este sistema também utiliza a linha de atraso para gerar o sinal de entrada, como demonstrado na figura 4.3. O ANFIS foi escolhido porque obteve bons resultados na previsão desta série (relatados no artigo original e constatado nos experimentos), e porque é incluído no *software* MATLAB.

Os dois últimos sistemas consistem nas mesmas arquiteturas, com a diferença que utilizam NDIs encadeados ao invés da linha de atraso, como ilustrados nas figuras 4.4 e 4.5. Os NDIs encadeados têm o papel de gerar aproximações da derivada

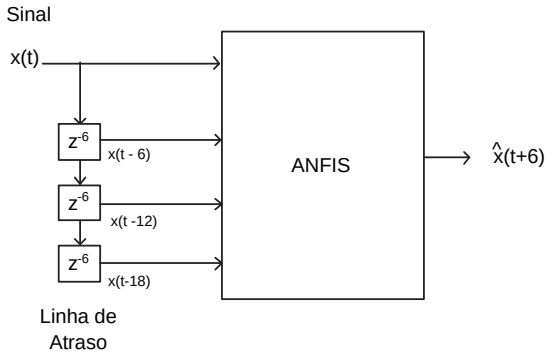

Figura 4.3: ANFIS com linhas de atraso para previsão da série de Mackey-Glass.

primeira, segunda e terceira do sinal original. No entanto, estas informações não são instantâneas, mas relativas a uma janela de tempo.

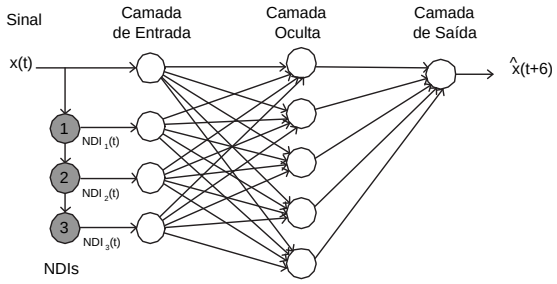

Figura 4.4: MLP com NDIs para previsão da série de Mackey-Glass.

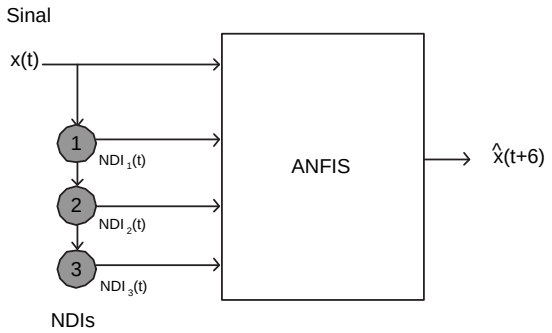

Figura 4.5: ANFIS com NDIs para previsão da série de Mackey-Glass.

Os NDIs utilizam médias móveis, portanto sua janela temporal não é facilmente definida. Deve-se escolher parâmetros μ e η de forma que o sinal resultante represente a mudança do sinal original ocorrida nos últimos instantes. Os valores utilizados para os NDIs 1, 2 e 3 foram, respectivamente, $\mu_1 = 0{,}9$, $\eta_1 = 0{,}8$, $\mu_2 = 0{,}9$, $\eta_2 = 0{,}2$, $\mu_3 = 0{,}9$ e $\eta_3 = 0{,}2$. Estes valores, de μ e η não foram representados na figura 3.6, mas a partir dela pode-se inferir que não há muita distinção entre dados recentes e remotos, todos possuem peso muito próximos. Os NDIs gerados, são portanto, bastante tolerantes a ruído e representam mudanças ocorridas em longos intervalos de tempo. Estas escolhas foram feitas para se obter informações relativas

a uma janela temporal igual ou maior que a janela temporal das linhas de atraso, utilizadas pelos dois primeiros sistemas.

Os parâmetros das arquiteturas foram escolhidos sem visar a busca por soluções ótimas, mas simplesmente para torná-las simples o suficiente para analisá-las e comparar suas formas de solução do problema.

4.1.3 Preparação dos Dados

Como as arquiteturas utilizam aprendizado em lote, foi gerado todo o conjunto de treinamento em memória antes da aplicação do aprendizado. Primeiramente, as mil amostras de teste e treinamento da seqüência temporal foram geradas, conforme já mencionado e delas foi subtraído a média de toda a seqüência (0,9194). Esta prática é comum no pré-processamento de sinais para facilitar o aprendizado em redes neurais.

O conjunto de treinamento para as duas primeiras arquiteturas é exatamente o mesmo, vetores do tipo:

$[x(t-18), x(t-12), x(t-6), x(t), x(t+6)]$,

onde os primeiros quatro valores correspondem aos dados de entrada, ou seja, o sinal original e o resultado da linha de atraso, e o último valor ($x(t+6)$) corresponde à resposta desejada (previsão da seqüência). Este conjunto de treinamento corresponde à mesma seqüência de dados utilizada em (JANG, 1993).

Para as duas últimas arquiteturas, aplicou-se a seqüência temporal à cadeia de NDIs e foram colecionadas suas respostas, gerando vetores do tipo:

$[x(t), NDI_1(t), NDI_2(t), NDI_3(t), x(t+6)]$,

Sendo os primeiros quatro valores, dados de entrada e o último ($x(t+6)$), a resposta desejada.

Os dois conjuntos de treinamento (somente com os valores de entrada) podem ser observados na figura 4.6. É importante ressaltar que, apesar de diferentes, os conjuntos de treinamento têm o mesmo número de dimensões. Logo, diferenças nos resultados de treinamento podem ser explicados somente pela diferença da natureza da informação gerada pelas linhas de atraso e pelos NDIs. Basicamente, as linhas de atraso representam o tempo no espaço e os NDIs representam o tempo sob a forma de mudança.

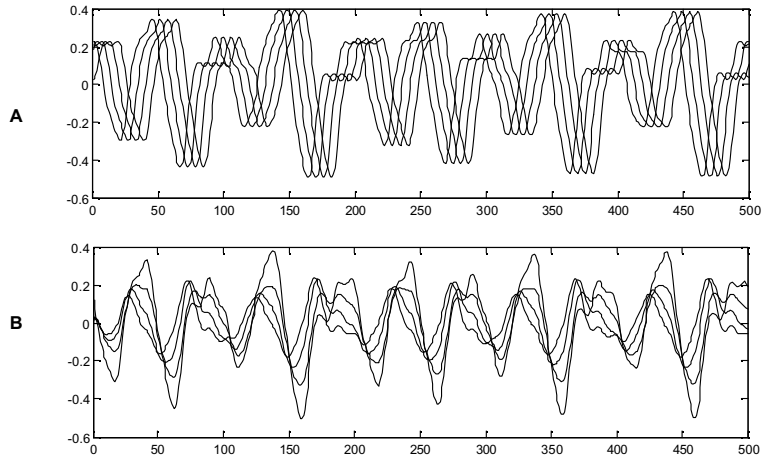

Figura 4.6: (A) Dados de entrada da série de Mackey-Glass gerados pela linha de atraso. (B) Dados de entrada da série de Mackey-Glass gerados pela cadeia de NDIs.

4.1.4 Metodologia de Testes

Para comparar os resultados dos quatro sistemas, utilizou-se a métrica NRMSE (do inglês *Normalized Root Mean Squared Error*), definida pela equação 4.2. Esta métrica de erro também foi utilizada em (GERS; ECK; SCHMIDHUBER, 2001) para comparar a eficiência de diversas arquiteturas neste problema de previsão.

$$NRMSE = \frac{< (y_k - t_k)^2 >^{\frac{1}{2}}}{< (t_k - < t_k >)^2 >^{\frac{1}{2}}} \tag{4.2}$$

A função $< . >$ corresponde a operação de média aritmética, y_k a saída obtida e t_k a saída desejada.

Como já foi dito, os experimentos foram realizados em MATLAB, utilizando o pacote de Redes Neurais e o pacote de Sistemas *Fuzzy* que acompanham este *software*, para implementar o *Perceptron de Múltiplas Camadas* e o ANFIS. Os parâmetros de treinamento utilizados foram os seguintes:

- Para o *Perceptron de Múltiplas Camadas* foi utilizado o aprendizado padrão TRAINLM (*Levenberg-Marquardt backpropagation*), conhecido por sua rapidez de convergência com arquiteturas simples. Os parâmetros para o TRAINLM foram: taxa de aprendizagem 0,01, *momentum* 0,0 e 500 épocas de treinamento. As funções de ativação dos neurônios ocultos e do neurônio de saída foram, respectivamente, função logística e função linear.

- O ANFIS foi gerado com os parâmetros predefinidos a fim de reproduzir os mesmos resultados relatados no artigo do autor (JANG, 1993).

Cada um dos testes relatados foi repetido pelo menos dez vezes e selecionado o resultado mais comum. A pequena quantidade de épocas de treinamento escolhida foi suficiente para constatar os resultados, pois o erro de previsão em qualquer uma das arquiteturas cai rapidamente e estabiliza já nas primeiras épocas de treinamento.

4.1.5 Resultados dos Experimentos

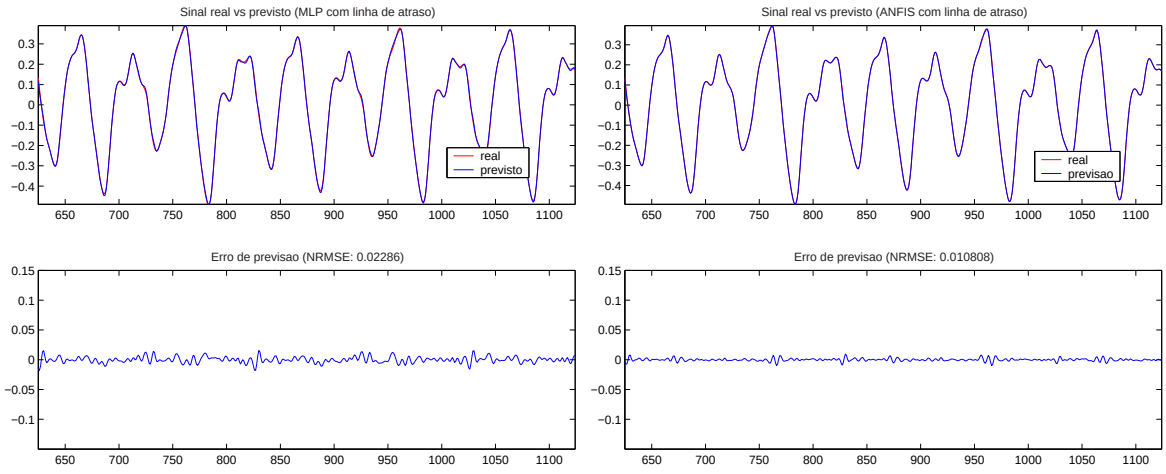

Figura 4.7: (Esquerda) Resultado do teste de previsão do MLP com linha de atraso para série de Mackey-Glass. (Direita) Resultado do teste de previsão do ANFIS com linha de atraso para série de Mackey-Glass.

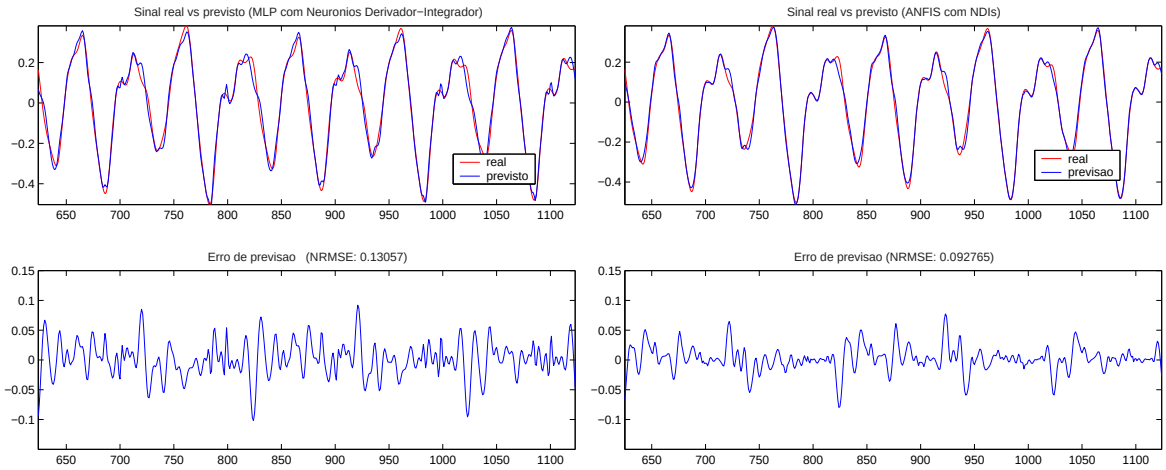

Figura 4.8: (Esquerda) Resultado do teste de previsão do MLP com NDIs encadeados para série de Mackey-Glass. (Direita) Resultado do teste de previsão do ANFIS com NDIs encadeados para série de Mackey-Glass.

Os resultados de previsão gerados pelo uso de NDIs (figura 4.8) foram inferiores aos obtidos pelo uso de linhas de atraso (figura 4.7). O NRMSE do MLP com linha de atraso foi de 0,02286 e com o NDI foi de 0,13057 e o NRMSE do ANFIS com linha de atraso foi de 0,01081 e com o NDI, 0,09277. Provavelmente estas diferenças provêm do fato de se tratar de uma seqüência caótica, que como já foi dito, depende de precisão para que haja boa previsão. Os NDIs perdem informação, pois realizam médias móveis. Pequenas diferenças de padrões de entrada são suprimidas pelas médias e levam a ambigüidades na representação temporal.

Para comprovar esta suposição, foram realizados mais dois experimentos alterando a seqüência temporal de treinamento. No primeiro experimento foi aplicado ruído ao sinal e no segundo, foram adicionadas constantes ao sinal, fazendo com que o sinal se deslocasse no eixo da amplitude. Para garantir que a rede reconhecesse estas perturbações como uma informação inútil para fins de previsão, o conjunto de treinamento foi triplicado, em ambos os experimentos, e as perturbações aplicadas em cada instância foram diferentes, ou seja, para cada ponto previsto, foram criados três pontos diferentes no espaço de entrada. E para compensar o aumento do conjunto de treinamento, foi divido por três o número de épocas de treinamento.

A seqüência de teste dos dois experimentos adicionais permaneceram iguais à seqüência de teste do problema original. Esperou-se com isso, medir a capacidade das arquiteturas de aprendizagem em obter mapeamentos que ignorassem as perturbações realizadas e generalizassem a solução o suficiente para prever os dados de teste que não sofreram das mesmas perturbações.

A figura 4.9 apresenta parte dos dados de entrada utilizados no primeiro experimento (tanto para o caso com linha de atraso como para o caso com cadeia de NDIs). A seqüência temporal foi adicionada de ruído de distribuição uniforme no intervalo $[-0{,}1; 0{,}1]$. O objetivo deste experimento foi o de verificar a sensibilidade em relação à qualidade dos dados de entrada para obter o correto mapeamento de previsão da série temporal.

A figura 4.10 apresenta o resultado de previsão para as primeiras duas arquiteturas que não utilizam NDI. Ambas obtiveram um aumento substancial no erro de previsão, alcançando NRMSE de 0,1, praticamente dez vezes maior. O MLP e

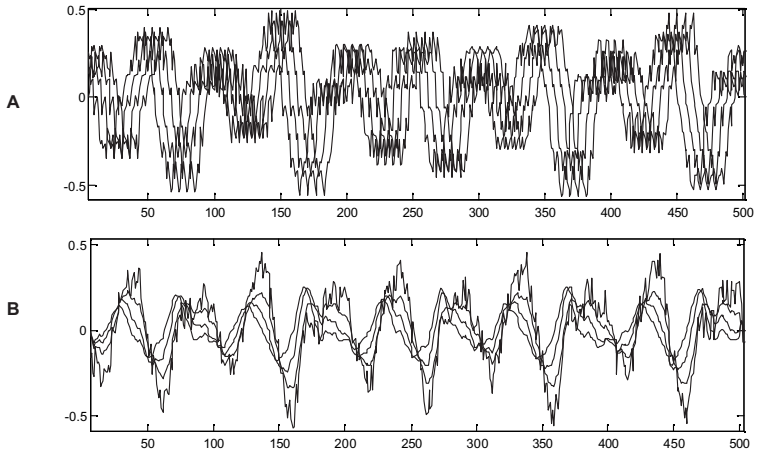

Figura 4.9: (A) Um terço dos dados de entrada com ruído da série de Mackey-Glass gerados pela linha de atraso. (B) Um terço dos dados de entrada com ruído da série de Mackey-Glass gerados pela cadeia de NDIs.

o ANFIS com NDIs, por outro lado, não foram tão prejudicados pelo ruído (figura 4.11). Seus NRMSEs chegaram a 0,16 e 0,14, respectivamente.

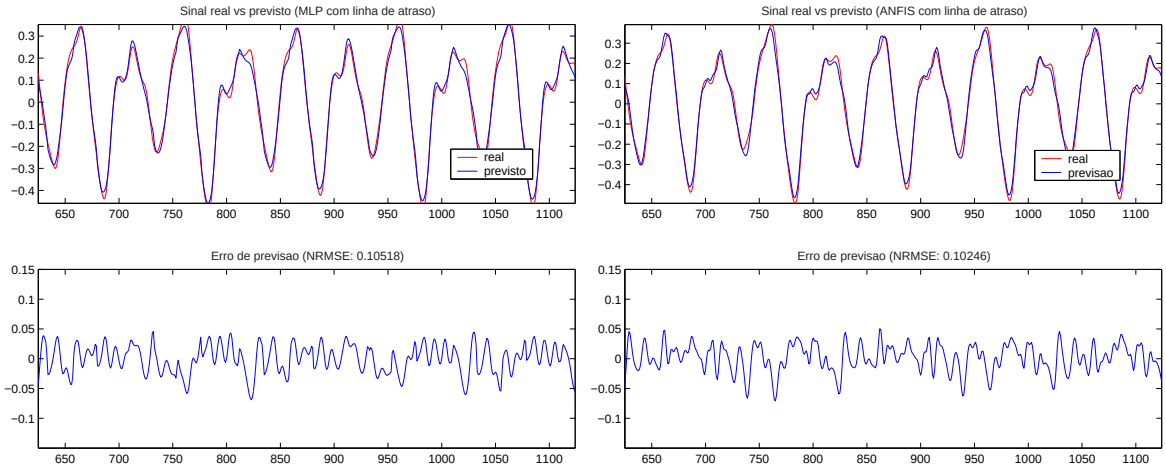

Figura 4.10: (Esquerda) Resultado do teste de previsão do MLP com linha de atraso para série de Mackey-Glass com ruído. (Direita) Resultado do teste de previsão do ANFIS com linha de atraso para série de Mackey-Glass com ruído.

Os resultados deste experimento comprovam a dependência da precisão dos dados para a realização da previsão. Tanto o MLP com linha de atraso, como o ANFIS aumentaram o erro de previsão para níveis próximos do MLP com NDIs. Este último, pouco sofreu com a perturbação, devido à filtragem do ruído realizada pelos NDIs. Supôs-se então, que os dois primeiros sistemas de aprendizagem estivessem memorizando um mapeamento preciso das entradas para a saída baseando-se nas informações absolutas dos valores de entrada. Para comprovar esta suposição, foi realizado o segundo experimento.

A figura 4.12 apresenta os dados utilizados no segundo experimento (tanto para o caso com linha de atraso como para o caso com cadeia de NDIs), gerado pela concatenação dos dados originais deslocados em $-0{,}5$, 0,5 e 1,0. Como os dados

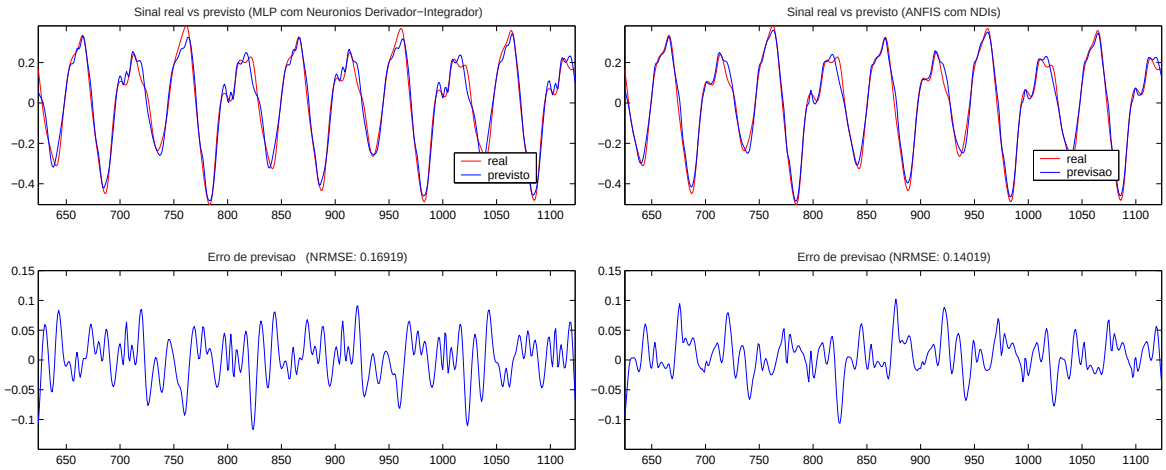

Figura 4.11: (Esquerda) Resultado do teste de previsão do MLP com NDIs encadeados para série de Mackey-Glass com ruído. (Direita) Resultado do teste de previsão do ANFIS com NDIs encadeados para série de Mackey-Glass com ruído.

de teste são diferentes dos dados de treinamento, as arquiteturas de aprendizado deveriam suprimir a informação absoluta dos valores da seqüência temporal e utilizar somente a relação entre os valores de entrada para obter sucesso neste experimento.

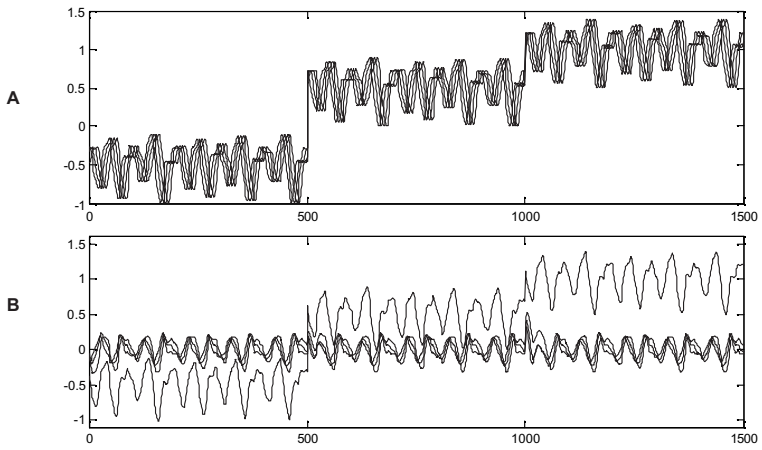

Figura 4.12: (A) Dados de entrada com deslocamento −0,5, 0,5 e 1,0 da série de Mackey-Glass gerados pela linha de atraso. (B) Dados de entrada com deslocamento −0,5, 0,5 e 1,0 da série de Mackey-Glass gerados pela cadeia de NDIs.

Pode-se perceber pela figura 4.13, que o MLP com linha de atraso e o ANFIS não foram capazes de generalizar a solução para deslocamentos diferentes daqueles que foram treinados. Em contrapartida, a figura 4.14 comprova a utilidade do NDI para a generalização correta deste problema.

Os resultados de todos os experimentos estão resumidos na tabela 4.1. Comparando os dados de entrada gerados pela linha de atraso com os dados dos NDIs (figuras 4.9 e 4.12), torna-se evidente que os NDIs geram informações com menos ruído e independente do deslocamento, razão pela qual, permitiu que as arquiteturas com NDIs solucionassem o problema de forma satisfatória nos dois casos. Esta conclusão reforça a preocupação apontada em (CLARK; THORNTON, 1997) quanto à dificuldade pela busca de representações internas para a solução de problemas e a im-

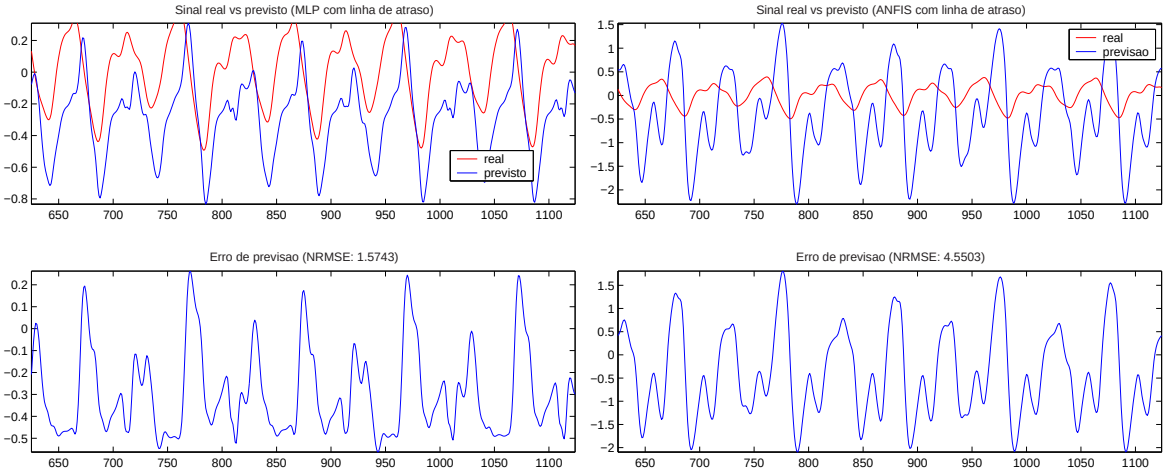

Figura 4.13: (Esquerda) Resultado do teste de previsão do MLP com linha de atraso para série de Mackey-Glass com deslocamento. (Direita) Resultado do teste de previsão do ANFIS com linha de atraso para série de Mackey-Glass com deslocamento.

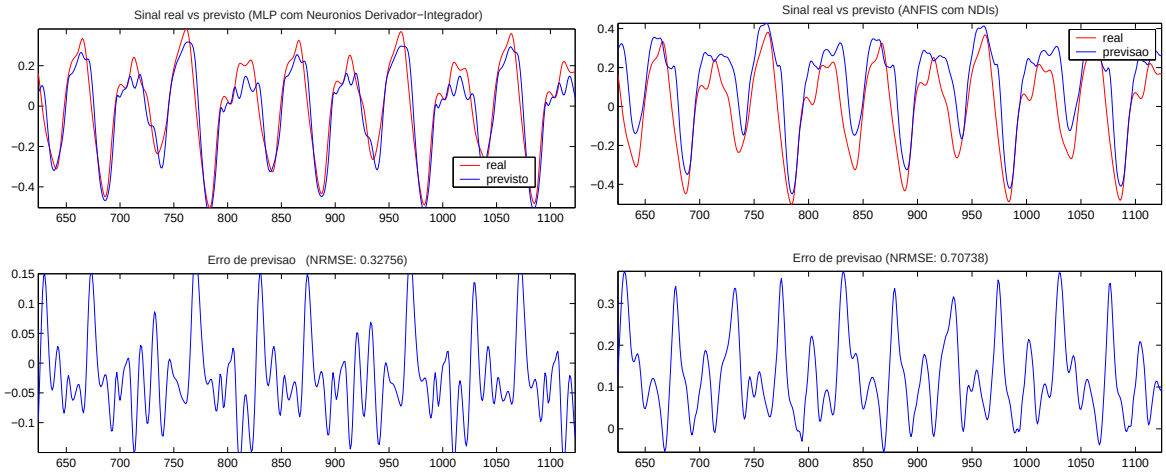

Figura 4.14: (Esquerda) Resultado do teste de previsão do MLP com NDIs encadeados para série de Mackey-Glass com deslocamento. (Direita) Resultado do teste de previsão do ANFIS com NDIs encadeados para série de Mackey-Glass com deslocamento.

portância pela escolha da representação dos dados de entrada e de saída adequadas em arquiteturas conexionistas.

4.2 Aplicação de Controle

As aplicações de controle tratam de problemas dinâmicos, onde geralmente é necessário manter o estado de um sistema físico em determinada faixa de valores. O controle é feito através de ações que afetam indiretamente o estado do sistema. Problemas de controle podem ser solucionados com diversas técnicas, dentre elas, o aprendizado por reforço.

Para ilustrar o uso de NDIs em aplicações de controle, escolheu-se um problema clássico de Inteligência Computacional, chamado *Pole Balancing* (BARTO; SUTTON; ANDERSON, 1983). Este problema consiste em um carrinho que se move

Tabela 4.1: Resumo dos experimentos com a série de Mackey-Glass

Experimento	Arquitetura	NRMSE Linhas de Atraso	NRMSE NDIs encadeados
Padrão	MLP	0,02288	0,13057
	ANFIS	0,01081	0,09277
Com Ruído	MLP	0,10518	0,16919
	ANFIS	0,10246	0,14019
Com Deslocamento	MLP	1,57430	0,32756
	ANFIS	4,55030	0,70738

somente em duas direções e tem como objetivo, manter equilibrado uma haste dentro de uma área limitada (figura 4.15). Se o carrinho sair da área delimitada ($-2{,}4 \leq x \leq 2{,}4$) ou deixar que a inclinação da haste ultrapasse 12 graus para qualquer um dos lados, é considerado que falhou na tentativa.

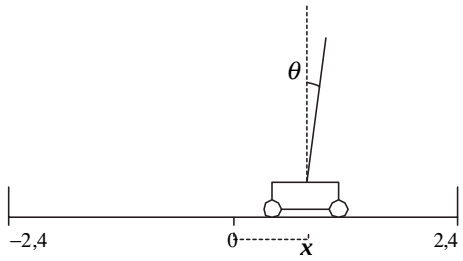

Figura 4.15: Problema *Pole Balancing*

O estado do carrinho é definido por quatro variáveis:

- x_t = posição horizontal do carrinho, relativo ao solo.
- $\dot{x}_t$ = velocidade horizontal do carrinho.
- θ_t = ângulo de inclinação da haste.
- $\dot{\theta}_t$ = velocidade angular da haste

As variáveis de estado são atualizadas a cada passo da simulação conforme as equações diferenciais 4.3-4.8:

$$x_{t+\tau} = x_t + \tau \dot{x}_t, \tag{4.3}$$

$$\dot{x}_{t+\tau} = \dot{x}_t + \tau \ddot{x}_t, \tag{4.4}$$

$$\theta_{t+\tau} = \theta t + \tau \dot{\theta}_t, \tag{4.5}$$

$$\dot{\theta}_{t+\tau} = \dot{\theta}_t + \tau \ddot{\theta}_t \tag{4.6}$$

$$\ddot{\theta}_t = \frac{-g \sin\theta_t + \cos\theta_t [\frac{-F_t - m_p l \dot{\theta}_t^2 \sin\theta t + \mu_c sgn(\dot{x}_t)}{m_c + m_p}] - \frac{\mu_p \dot{\theta}_t}{m_p l}}{l[\frac{4}{3} - \frac{m_p \cos^2\theta_t}{m_c + m_p}]}, \tag{4.7}$$

$$\ddot{x}_t = \frac{F_t + m_p l[\dot{\theta}_t^2 \sin\theta_t - \ddot{\theta}_t \cos\theta_t] - \mu_c sgn(\dot{x}_t)}{m_c + m_p} \tag{4.8}$$

onde: $m_c = 1{,}0$ = massa do carrinho em quilos, $m_p = 0{,}1$ = massa da haste em quilos, $l = 0{,}5$ = centro de massa da haste até a base em metros, $\mu_c = 0{,}0005$ = coeficiente de fricção do carrinho no solo, $\mu_p = 0{,}000002$ = coeficiente de fricção da haste, $g = -9{,}8$ = aceleração da gravidade em metros por segundo e $\tau = 0{,}02$ = passo da simulação em segundos. A ação é dada pela força lateral aplicada ao carrinho (F_t), que pode ser de 10 ou -10 newtons.

Este problema também foi abordado em (ANDERSON, 1986). A solução sugerida no trabalho envolve o uso de *Perceptrons de Múltiplas Camadas* com aprendizado por reforço, como ilustrado na figura 4.16. Existem duas redes, MLP_1 e MLP_2. A rede MLP_1 faz o papel do crítico adaptativo, ou seja, tem o objetivo de prever o valor do estado do carrinho. Esta rede é treinada com o reforço provindo do simulador. O reforço é zero enquanto o carrinho estiver em um estado válido e torna-se -1 ao atingir um estado de falha. A rede MLP_2 decide pela ação que o carrinho deve tomar a cada instante. Esta rede é treinada com o sinal de reforço previsto da MLP_1. Ambas as redes recebem como entrada, o estado do carrinho $(x, \theta, \dot{x}, \dot{\theta})$.

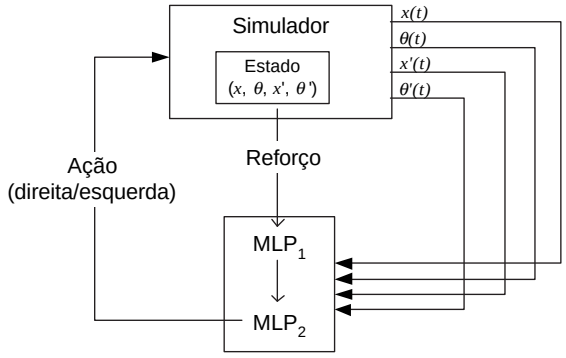

Figura 4.16: Arquitetura conexionista padrão aplicada ao problema *Pole Balancing*

A arquitetura descrita foi implementada em MATLAB com os mesmos parâmetros do trabalho original. O treinamento também seguiu as descrições do autor. Trata-se de um tipo de aprendizado por retropropagação de erro incremental que usa o sinal de reforço como o sinal de erro a minimizar. Cada tentativa do carrinho começa em um estado inicial aleatório ($x, \theta, \dot{x}, \dot{\theta}$ contidos, respectivamente, em $[-2{,}3; 2{,}3]$, $[-11{,}4; 11{,}4]$, $[-0{,}25; 0{,}25]$ e $[-0{,}25; 0{,}25]$). A tentativa encerra quando o carrinho atinge um estado de falha, gerando o sinal de reforço -1. A medida que o treinamento vai sendo aplicado, o tempo de equilíbrio (duração das tentativas) aumenta.

A figura 4.17 apresenta o gráfico do treinamento em 8000 tentativas com esta arquitetura. Para não comprometer o tempo de simulação, limitou-se o tempo de cada tentativa em 5000. Pode-se perceber pela figura que o aprendizado obteve sucesso, alcançando na maioria das tentativas o valor máximo de duração. As tentativas que não foram bem sucedidas no final do treinamento são impossíveis de solucionar, pois seus estados iniciais não permitem recuperação do equilíbrio da haste.

O problema foi resolvido com facilidade pela arquitetura. No entanto, foi necessário fornecer informações estáticas e dinâmicas (derivadas). Em robótica, as informações dinâmicas não são comuns. Geralmente os sensores de robôs medem informações estáticas a cada instante de tempo, como a distância a outros objetos, a luminosidade e forma de objetos, a colisão com obstáculos, o ângulo das juntas

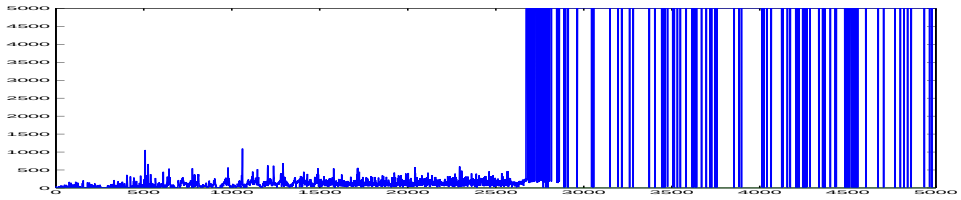

Figura 4.17: Duração das tentativas durante treinamento com arquitetura padrão.

mecânicas, etc. Para que as soluções deste problema teórico sejam úteis no treinamento de robôs reais, é necessário eliminar as informações dinâmicas ou extraí-las diretamente da informação estática.

Antes de tentar substituir as derivadas neste problema, é importante verificar a real necessidade por este tipo de informação dinâmica. A arquitetura foi modificada conforme a figura 4.18, simplesmente omitindo as derivadas para as redes neurais e treinando com os mesmos parâmetros.

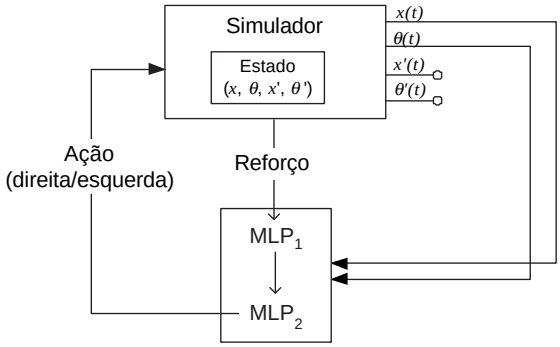

Figura 4.18: Arquitetura conexionista simplificada aplicada ao problema *Pole Balancing*

Os resultados do treinamento (figura 4.19) deixam claro que o problema não pode ser resolvido sem a informação dinâmica. Existe aprendizado, mas o carrinho não ultrapassa o limite de 800 instantes de tempo equilibrando.

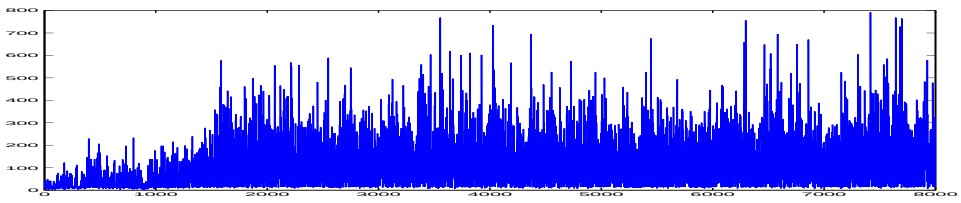

Figura 4.19: Duração das tentativas durante treinamento com arquitetura simplificada.

Para substituir a informação temporal das derivadas, novamente tentou-se aplicar linhas de atraso, por ser um recurso comumente utilizado na Inteligência Computacional. A linha de atraso foi aplicada sobre os sinais estáticos de posição e ângulo (x e θ) para criar uma representação temporal duplicada no espaço, tal como ilustrado na figura 4.20.

A arquitetura foi testada com linhas de atraso de 1, 2 e 3 instantes de tempo de atraso e em nenhum dos casos se obteve sucesso. A figura 4.21 apresenta o

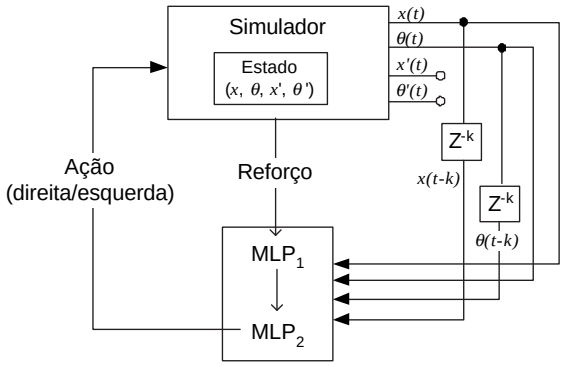

Figura 4.20: Arquitetura conexionista com linha de atraso aplicada ao problema *Pole Balancing*

treinamento em 20000 tentativas com linhas de atraso de uma unidade de tempo. Os resultados foram ligeiramente melhores que o caso sem derivadas, mas não foram suficientes para manter a haste equilibrado por mais de 1500 instantes de tempo.

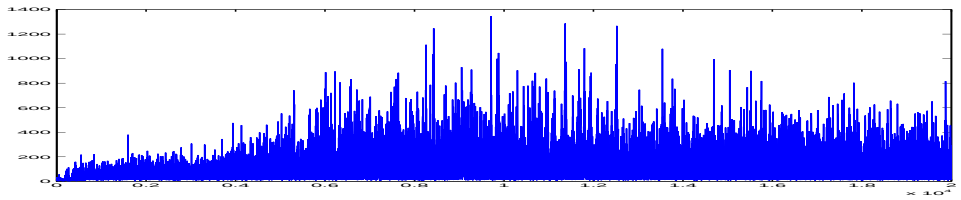

Figura 4.21: Duração das tentativas durante treinamento com arquitetura com linha de atraso.

Por último, aplicou-se ao invés de linhas de atraso, NDIs às informações estáticas (figura 4.22). Como o NDI gera informação de mudança, esperou-se obter informações da mesma natureza que as derivadas suprimidas $\dot{x}$ e $\dot{\theta}$. Primeiramente, utilizou-se NDIs com parâmetros $\eta = 0{,}0$ e $\mu = 0{,}0$, para simplesmente diferenciar o sinal. A figura 4.23 mostra os resultados deste treinamento. O NDI recriou a informação dinâmica necessária para a solução do problema.

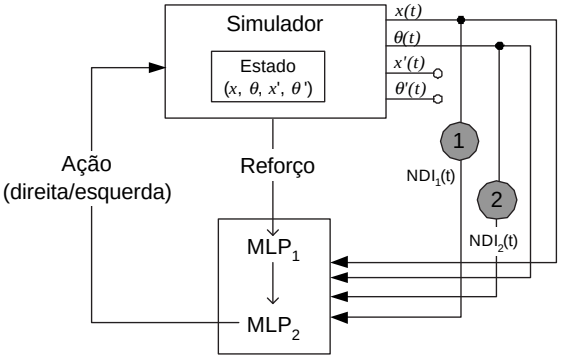

Figura 4.22: Arquitetura conexionista com NDIs aplicada ao problema *Pole Balancing*

Após o sucesso com a diferenciação, foi feita uma seqüência de testes variando os parâmetros de η e μ para identificar outras combinações de parâmetros que tam-

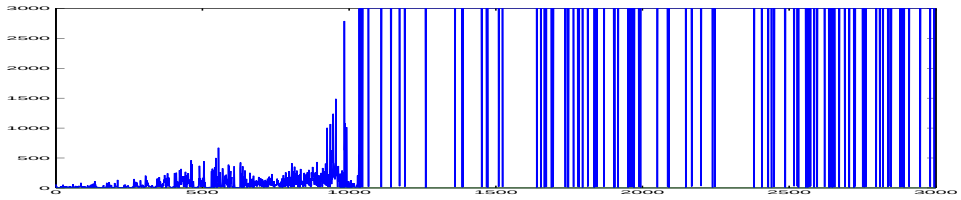

Figura 4.23: Duração das tentativas durante treinamento com arquitetura de diferenciação.

bém fossem bem sucedidas. O objetivo destes testes era descobrir parâmetros que determinassem NDIs tolerantes a ruído capazes de resolver o problema original e outros cujos valores de entrada (x e θ) fossem poluídos com ruído. No entanto, a bateria de testes provou que o problema é muito sensível a pequenas variações do sinal de entrada e exige resposta instantânea do sistema, impossibilitando o uso de NDIs tolerantes a ruído. A figura 4.24 apresenta o treinamento pelo uso de NDIs com $\eta = 0{,}6$ e $\mu = 0{,}7$. Nota-se que houve momentos do treinamento onde se encontrou soluções boas, mas não permaneceram estáveis durante o treinamento e se perderam. Neste experimento foi necessário multiplicar a saída dos NDIs por5 para compensar a perda de amplitude do sinal.

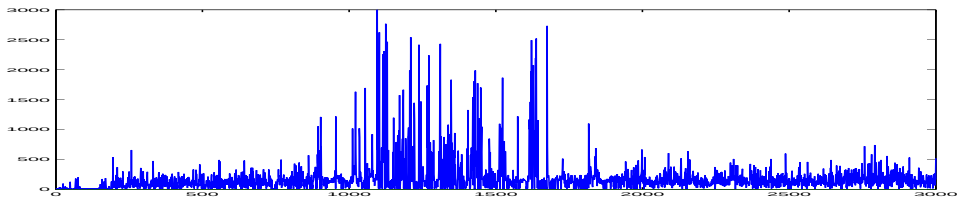

Figura 4.24: Duração das tentativas durante treinamento com arquitetura de NDIs.

Este problema teórico indica que informações dinâmicas podem ser fundamentais nas soluções de robótica e podem ser extraídas de informações estáticas com o uso de NDIs. Acredita-se que os sinais sensoriais reais sejam mais redundantes e mais ruidosos que no problema teórico. Além disso, a resposta do robô em cada instante de tempo não é tão específica quanto a resposta exigida pelo problema*Pole Balancing*. A solução encontrada pela arquitetura original é um movimento que oscila para a direita e esquerda em alta freqüência, fazendo com que se estabilize a posição da haste. Esta solução é impraticável em sistemas reais, uma vez que os motores não respondem instantaneamente aos comandos e possuem inércia. Estas duas circunstâncias tipicamente encontradas na robótica indicam que o NDI possa ter utilidade nos problemas desta área de pesquisa.

4.3 Segmentação de Seqüências Temporais

No capítulo 2, foram analisadas diversas arquiteturas de segmentação que geram representações para trechos de seqüências temporais baseadas em mapas auto-organizáveis. Nesta seção são apresentados os resultados da utilização de NDIs na entrada de um SOM e de um ARAVQ, para demonstrar a influência na segmentação do fluxo sensorial e comparar os resultados obtidos pela aplicação do mesmo sinal temporal "8" relatado no capítulo 2.

O experimento com o SOM foi realizado nas mesmas condições descritas no capítulo 2: um mapa unidimensional circular de 20 neurônios, com parâmetros$\delta = 1$ reduzido linearmente até zero e $\sigma = 2$ linearmente reduzido até 1 e 10 apresentações da seqüência temporal "8". No entanto, foram adicionados aos dados de entrada, os sinais gerados por NDIs associados a cada dimensão do padrão de entrada, como ilustrado na figura 4.25.

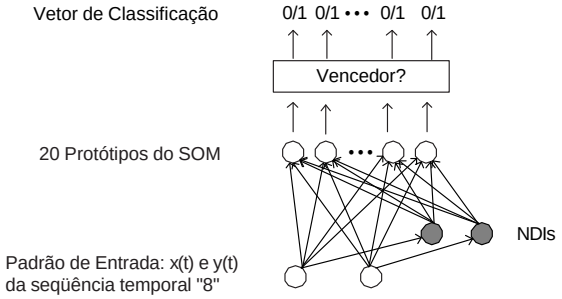

Figura 4.25: Aplicação de NDIs à entrada de um SOM padrão.

Os parâmetros para ambos os NDIs foram de $\eta = 0{,}7$ e $\mu = 0{,}7$ com a saída multiplicada por 10 para compensar a redução de amplitude do sinal e aumentar a importância da informação de mudança gerada pelos NDIs em relação a informação espacial dos padrões de entrada, no cálculo do erro de quantização do SOM. O resultado dos pesos relativos ao espaço de entrada original é mostrado no gráfico superior da figura 4.26. O gráfico inferior da mesma figura apresenta a saída do SOM para cada instante da seqüência temporal. Não existe ambiguidade na representação, mesmo no ponto de cruzamento da seqüência ($t = 50$ e 150). Esta arquitetura obteve o melhor resultado comparado com as outras soluções apresentadas no capítulo 2, pois em 96% dos experimentos houve convergência estável de representação.

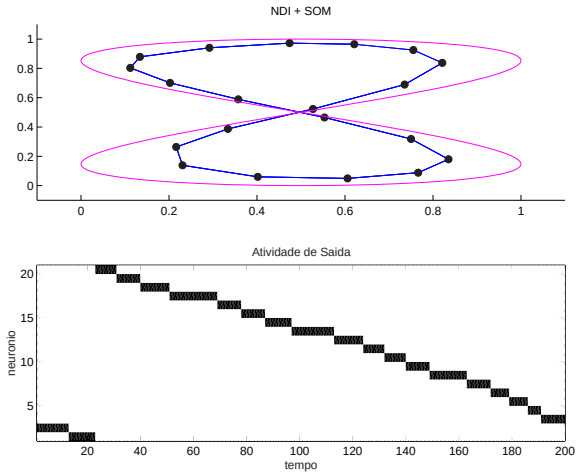

Figura 4.26: Superior: resultado dos protótipos do SOM com NDIs após 10 apresentações da seqüência temporal "8". Inferior: saída do SOM com NDIs para cada ponto da seqüência temporal.

Para verificar a estabilidade da representação gerada, após treinado o SOM, aplicou-se seqüências temporais "8" com perturbações de ruído, translação, deformação nas dimensões x e y e rotação. A classificação gerada para cada instante

da seqüência idealmente deveria ser a mesma. Verificou-se que o SOM com NDIs é bastante tolerante a todas as perturbações testadas, principalmente a perturbações de ruído e translação da seqüência. As figuras 4.27 e 4.28 apresentam os resultados de classificação da seqüência com ruído de distribuição uniforme nos intervalos $[-0{,}1; 0{,}1]$ e $[-0{,}2; 0{,}2]$, respectivamente. A figura 4.29 apresenta o resultado da perturbação de translação. A figura 4.30 apresenta o resultado da perturbação de deformação de eixos e por último, a figura 4.31 apresenta o resultado da perturbação de rotação do sinal.

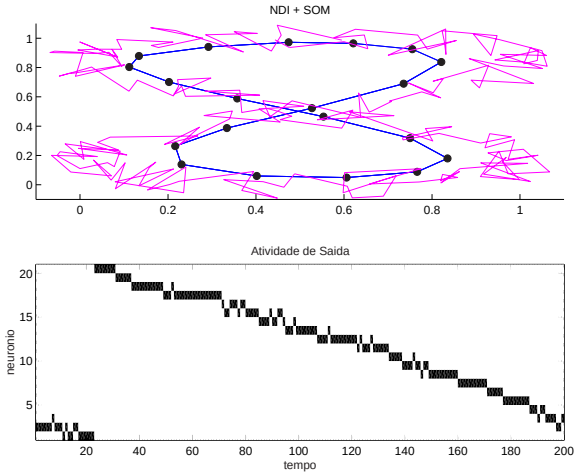

Figura 4.27: Superior: seqüência temporal "8" com ruído de distribuição uniforme no intervalo $[-0{,}1; 0{,}1]$ aplicada ao SOM com NDIs. Inferior: saída do SOM com NDIs para cada ponto da seqüência temporal.

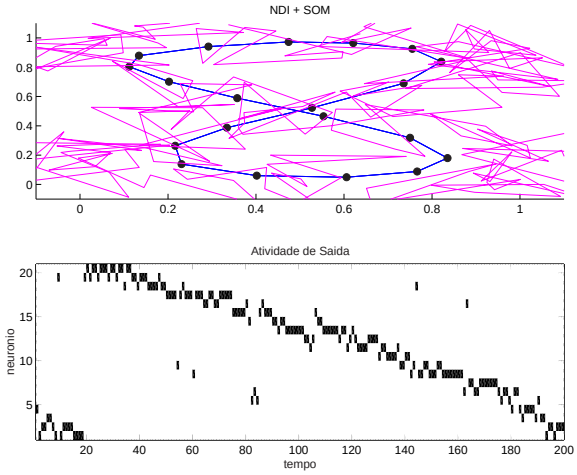

Figura 4.28: Superior: seqüência temporal "8" com ruído de distribuição uniforme no intervalo $[-0{,}2; 0{,}2]$ aplicada ao SOM com NDIs. Inferior: saída do SOM com NDIs para cada ponto da seqüência temporal.

No experimento com o ARAVQ também se utilizou NDIs para aumentar a dimensionalidade dos dados de entrada. Os parâmetros escolhidos foram: $\alpha = 0{,}05$, $\delta = 0{,}5$, $\epsilon = 0{,}2$ e $n = 10$ para o ARAVQ, e para os NDIs, $\eta = 0{,}7$ e $\mu = 0{,}7$, com a saída multiplicada por 18 para compensar a redução de amplitude do sinal. Os resultados de segmentação da seqüência temporal em forma de "8" são mostrados

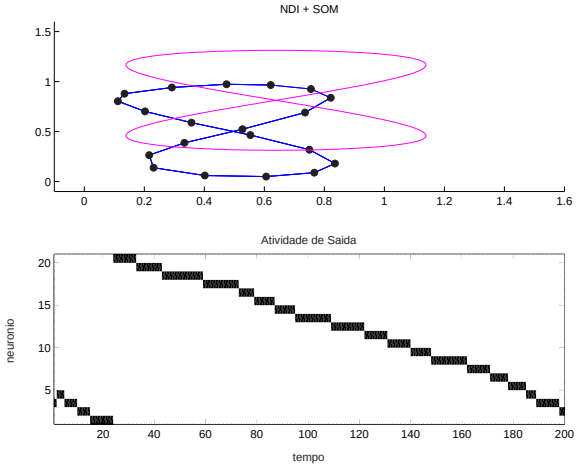

Figura 4.29: Superior: seqüência temporal "8" com deslocamento aplicada ao SOM com NDIs. Inferior: saída do SOM com NDIs para cada ponto da seqüência temporal.

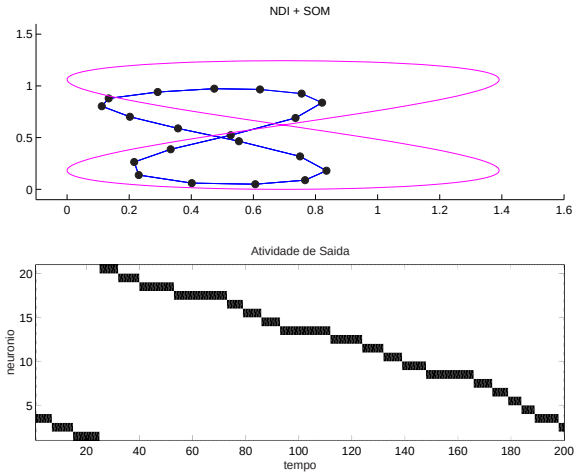

Figura 4.30: Superior: seqüência temporal "8" com deformação nos eixos da seqüência aplicada ao SOM com NDIs. Inferior: saída do SOM com NDIs para cada ponto da seqüência temporal.

na figura 4.32-A. Não houve nenhum tipo de seleção exaustiva de parâmetros para este experimento. O objetivo foi simplesmente de observar o tipo de transformação de representação que o NDI realiza e verificar a sua potencialidade. Outros valores para η e μ geram resultados semelhantes.

O resultado da segmentação relativo ao espaço de entrada original (figura 4.32-A) distingue o ponto de cruzamento $(0{,}5; 0{,}5)$ da seqüência temporal com dois protótipos (grupos 1 e 5), diferentemente do ARAVQ original (figura 2.38). Isto ocorre porque o ARAVQ está segmentando a seqüência temporal em conjunto com outro espaço de representação, no caso, o espaço gerado pelos NDIs, ilustrado na figura 4.32-B. Os grupos 1 e 5 encontram-se em locais distantes neste segundo espaço, o que torna possível sua distinção.

A aplicação de ruído sobre a seqüência original não prejudica a classificação final, como pode ser observado na figura 4.33. São apresentadas duas instâncias da seqüência temporal somadas com ruído uniforme de distribuição uniforme nos

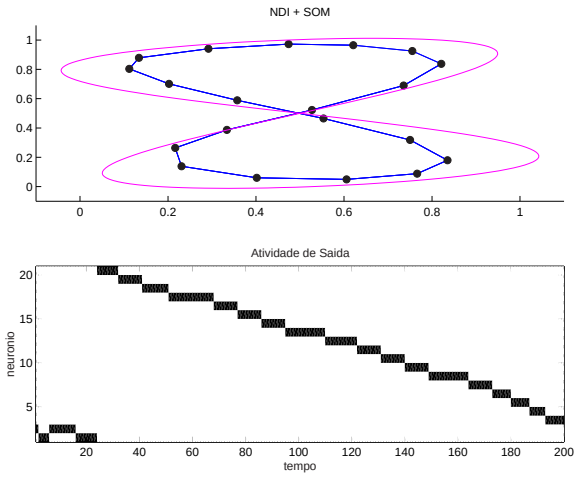

Figura 4.31: Superior: seqüência temporal "8" com rotação da seqüência aplicada ao SOM com NDIs. Inferior: saída do SOM com NDIs para cada ponto da seqüência temporal.

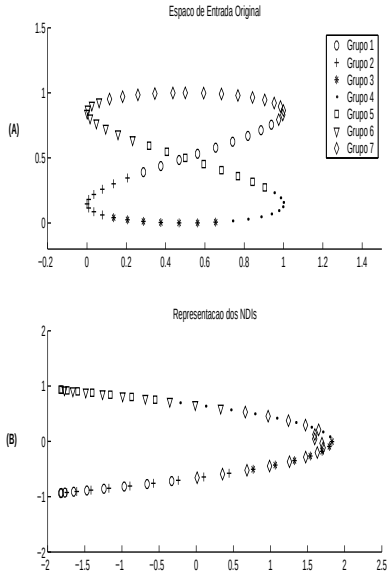

Figura 4.32: Resultado da classificação com ARAVQ e NDI da seqüência temporal "8" original.

intervalos $[-0{,}1; 0{,}1]$ e $[-0{,}2; 0{,}2]$. A propriedade de redução de ruído por integração temporal do NDI e pela média móvel do ARAVQ garante invariância na segmentação da seqüência.

A perturbação de translação no sinal, ilustrada na figura 4.34-A, gera pequenos erros na classificação final. A figura 4.34-B apresenta o espaço de representação gerado pelos NDIs e pode-se constatar que é praticamente invariável e semelhante à representação sem translação da figura 4.32-B.

A rotação é uma perturbação que afeta a representação do NDI, como pode ser observada na figura 4.35-B. No entanto os resultados de classificação finais (figura 4.35-A), ainda assim podem ser considerados satisfatórios, pois a maior parte dos pontos da seqüência temporal, permanece com a mesma classificação da seqüência original (figura 4.32-A).

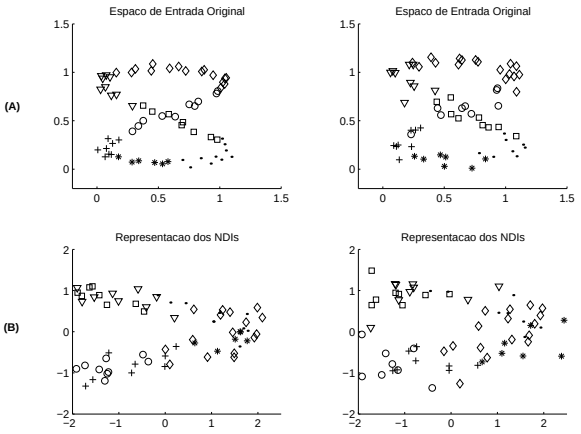

Figura 4.33: Resultado da classificação com ARAVQ e NDI da seqüência temporal "8" com ruído.

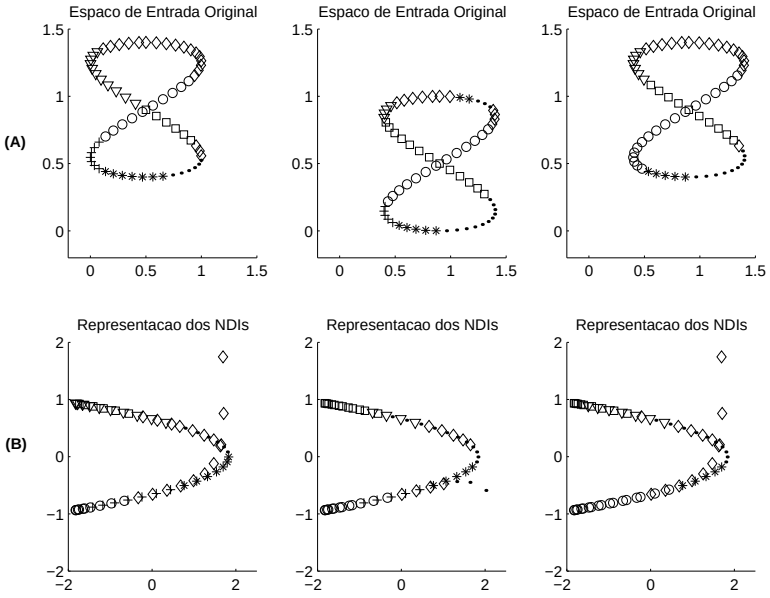

Figura 4.34: Resultado da classificação com ARAVQ e NDI da seqüência temporal "8" com translação.

Por último, alterou-se a magnitude do sinal. Esta perturbação, assim como a rotação, também prejudicou a classificação do ARAVQ com NDIs. Mesmo assim, pode-se observar na figura 4.36-B, que o espaço de representação gerado pelos NDIs é mais estável que o espaço de entrada (figura 4.36-A), provando que os NDIs reduzem a influência desta perturbação, aumentando a estabilidade da classificação.

Os testes realizados comprovam a utilidade dos NDIs aplicados a arquiteturas de segmentação de fluxo sensorial, pois garantem estabilidade de representação frente a perturbações comumente encontradas nestes sinais. No entanto, a informação gerada passa a ter um caráter dinâmico, ou seja, baseado na mudança do sinal. Esta característica pode ser desejável em determinados tipos de problemas. Cabe ao projetista da rede reconhecer o tipo de informação necessária para cada problema e aplicar a solução adequada. No caso de fluxos sensório-motores de robôs, a segmentação provavelmente deve ser feita tanto utilizando os dados estáticos (leitura direta dos sensores) como utilizando os dados dinâmicos (uso de NDIs), exatamente

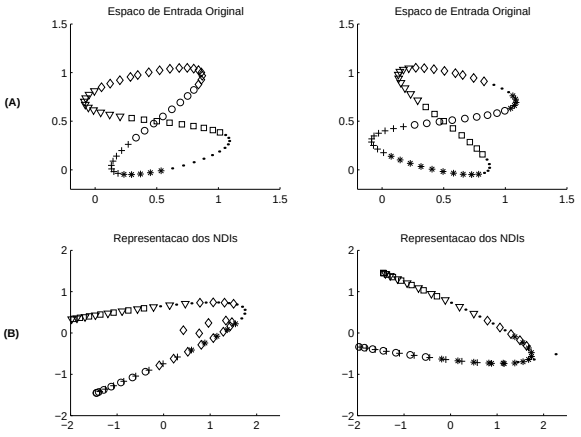

Figura 4.35: Resultado da classificação com ARAVQ e NDI da seqüência temporal "8" com rotação.

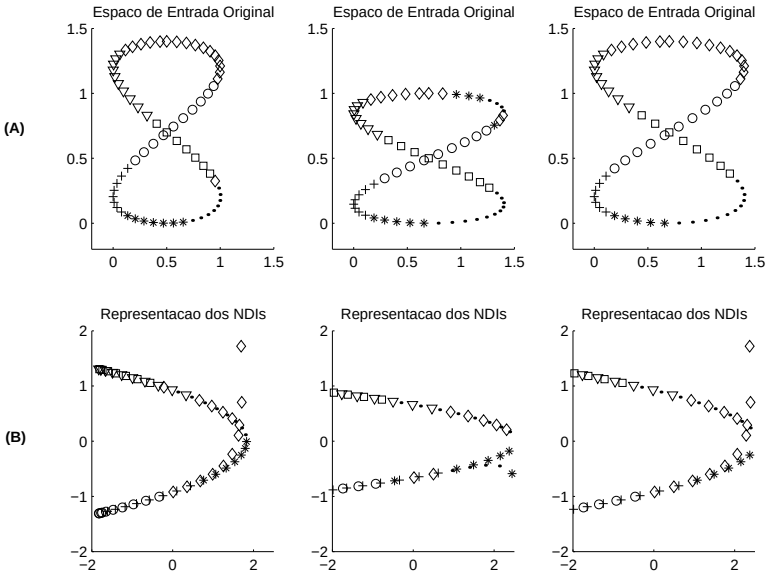

Figura 4.36: Resultado da classificação com ARAVQ e NDI da seqüência temporal "8" com deformação.

como foi feito nos experimentos teóricos. Desta forma, a cada instante se obtém uma descrição compacta do estado e da ação do robô.

Para fins de ilustração, utilizou-se o simulador do robô Khepera em MATLAB (NILSSON, 2001) para produzir os sinais temporais de dois dos oito sensores de proximidade do robô (figura 4.37) ao fazê-lo se locomover por um labirinto utilizando o comportamento de *following wall*. A seqüência coletada corresponde a um trecho da rota do robô, ilustrado na figura 4.38.

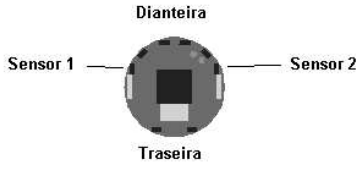

Figura 4.37: Sensores do Khepera usados para captura de sinal temporal.

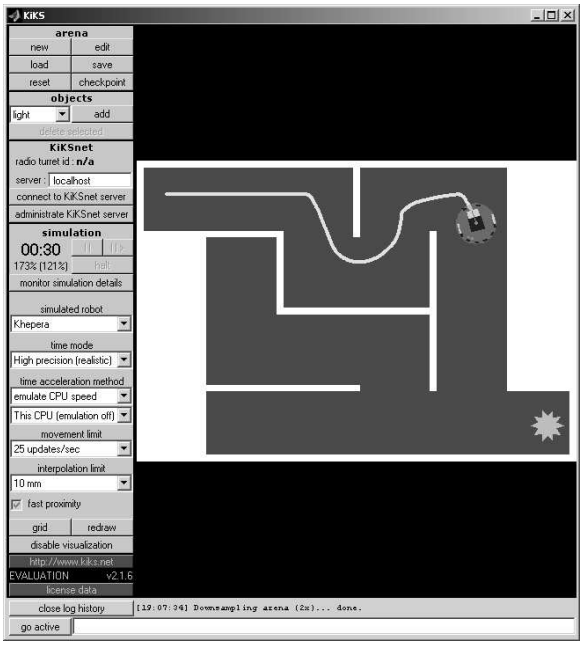

Figura 4.38: Rota produzida pelo Khepera para análise dos sinais sensoriais.

Os sinais temporais obtidos pelos dois sensores são ilustrados na figura 4.39. Nesta figura também são plotados os sinais gerados por NDIs com$\eta = 0{,}7$ e $\mu = 0{,}7$ associados a cada um dos sensores. A saída dos NDIs está multiplicada por 10. Pode-se perceber que as mudanças dos sinais sensoriais são representadas pelos NDIs com saltos de atividade, tanto positivos como negativos e a informação é propagada por alguns instantes de tempo. Este comportamento pode ser de grande valia para identificação de eventos importantes, como aproximação de objetos, mudanças entre corredor e curvas, oclusão de objetos, etc.

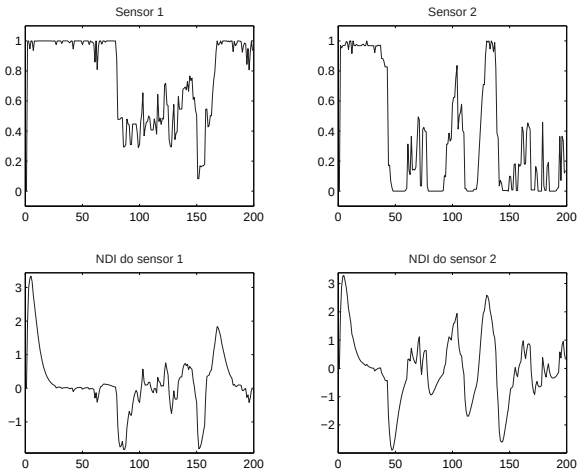

Figura 4.39: Sinais sensoriais capturados pelo Khepera e aplicação de NDIs para geração de representação temporal.

5 CONCLUSÕES

O trabalho abordou diferentes modelos que geram representação de dados adequada para a resolução de problemas que envolvem o tempo, ou seja, problemas onde a ordem dos dados carrega informação. Após a revisão, foi introduzida uma forma alternativa de representar os dados no tempo, uma forma que expressa mudança no tempo. O modelo proposto foi o de um neurônio que tem a funcionalidade de derivação e integração, dependendo dos parâmetros escolhidos. Constatou-se, pelos experimentos comparativos, que esta abordagem, além de prática por ser facilmente adaptável às arquiteturas conhecidas, mostrou-se muito eficaz para a resolução de determinados problemas temporais.

5.1 Análise dos resultados

Como visto no capítulo 4, o modelo neural introduzido resultou em baixo custo de implementação e de computação, rendendo a determinados problemas, soluções mais genéricas que as encontradas na literatura. É importante ressaltar, no entanto, que o modelo é somente mais uma alternativa para a representação temporal e deve ser usado quando julgado necessário, pois pode, em determinados problemas, com determinados parâmetros, simplesmente não ajudar na solução ou até prejudicar os resultados, assim como qualquer modelo mal utilizado da Inteligência Computacional.

Identificou-se que os problemas temporais que possuem alta correlação temporal (ex.: sinais sensoriais de robôs), com ou sem ruído, provavelmente podem se aproveitar das informações geradas pelo NDI. Em contrapartida, problemas com baixa correlação temporal ou simbólicos, (ex.: cadeias de DNA) provavelmente se agravariam com a utilização do neurônio.

As fraquezas do modelo se concentram sobre as propriedades de filtragem do neurônio. O neurônio pode ser visto como dois filtros passa-baixa de ordem um ligados em série e como visto no capítulo 3, existe uma grande alteração nas freqüências e pouca separação (em termos de freqüência de corte). Portanto, seu raio de atuação é limitado a determinadas freqüências. Seria possível substituir o modelo do neurônio por filtros de ordens maiores que um obtendo características mais desejáveis, sob o ponto de vista de filtragem. No entanto, optou-se por um modelo de poucos parâmetros, mais biologicamente plausível, de menor custo computacional e que ainda assim na saída representasse a informação de mudança com tolerância a ruídos.

5.2 Trabalhos futuros

O paralelo realizado neste trabalho entre o modelo neural NDI com os filtros digitais abre espaço para um estudo aprofundado sobre suas propriedades, as conseqüências sobre os dados processados e as limitações reais impostas pelo modelo, tanto no domínio freqüência como no domínio tempo, haja visto o comportamento de perda com o encadeamento em série destes neurônios.

O modelo genérico sugerido é parametrizável. A influência de seus parâmetros sobre o processamento foi analisada e relatada para facilitar a escolha dos parâmetros em problemas diversos. No entanto, seria de grande valia um algoritmo ou heurística de otimização destes valores para ser calculado em tempo real. Desta forma seria possível aplicar o neurônio em camadas intermediárias de um perceptron de múltiplas camadas e incluí-lo nas equações de treinamento do tipo retropropagação de erros.

Os testes simples realizados comprovaram o potencial do modelo. No entanto, os experimentos certamente não foram exaustivos. Muitas outras aplicações poderiam ser testadas com este neurônio, como aplicações de controladores inteligentes, visão de máquina, locomoção de robôs, etc.

Por fim, a pesquisa realizada sobre as formas de representação temporal existentes indica que se trata de uma área de estudos ainda pouco explorada. É necessário aprofundar a análise sobre os modelos existentes e possivelmente aprimorá-los para o uso em aplicações complexas como as da robótica.

REFERÊNCIAS

ANDERSON, C. W. **Learning and Problem Solving with Multilayer Connectionist Systems**. 1986. Tese (Doutorado em Ciência da Computação) — University of Massachusetts, Department of Computer and Information Science.

ARBIB, M. A. **Handbook of Brain Theory and Neural Networks** Massachusetts: MIT Press, 1995.

BAKKER, B.; LINåKER, F.; SCHMIDHUBER, J. Reinforcement Learning in Partially Observable Mobile Robot Domains Using Unsupervised Event Extraction. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, IROS, 2002, San Francisco. **Proceedings...** Piscataway: IEEE Press, 2002. v.3, p.190–196.

BARTO, A. G.; SUTTON, R. S.; ANDERSON, C. W. Neuronlike adaptive elements that can solve difficult learning control problems. **IEEE Transactions on systems, man and cybernetics**, [S.l.], v.SMC-13, p.834–846, 1983.

BOX, G. E. P.; JENKINS, G. M.; REINSEL, G. C. **Time series analysis**: forecasting and control. 3rd ed. Englewood Cliffs: Prentice-Hall, 1994.

CARPENTER, G. A.; GROSSBERG, S. A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine. **Computer Vision, Graphs and Image Processing**, [S.l.], v.37, p.54–115, 1987.

CARPENTER, G. A.; GROSSBERG, S.; ROSEN, D. B. Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system. **Neural Networks**, New York, v.4, n.6, p.759–772, Dec. 1991.

CHAPPELL, G. J.; TAYLOR, J. G. The Temporal Kohonen Map. **Neural Networks**, [S.l.], v.6, p.441–445, 1993.

CLARK, A.; THORNTON, C. Trading Spaces: computation, representation, and the limits of uninformed learning. **Behavioral and Brain Sciences**, [S.l.], v.20, p.57–90, 1997.

DONG, D. W.; ATICK, J. J. Statistics of Natural Time-Varying Images. **Network Computation in Neural Systems**, [S.l.], v.6, n.3, p.345–358, 1995.

DONG, D. W.; ATICK, J. J. Temporal decorrelation: a theory of lagged and nonlagged responses in the lateral geniculate nucleus. **Network Computation in Neural Systems**, [S.l.], v.6, p.159–178, 1995.

ELMAN, J. L. Finding Structure in Time. **Cognitive Science**, [S.l.], v.14, n.2, p.179–211, Apr. 1990.

ELMAN, J. L. Learning and Development in Neural Networks: the importance of starting small. **Cognition**, [S.l.], v.48, n.1, p.71–79, 1993.

EULIANO, N. R.; PRINCIPE, J. C. Spatio-temporal self-organizing feature maps. In: INTERNATIONAL CONFERENCE ON NEURAL NETWORKS, ICNN, 1996. **Proceedings...** New York: IEEE, 1996. v.4, p.1900–1905.

EULIANO, N. R.; PRINCIPE, J. C. A Spatio-Temporal Memory Based on SOMs with Activity Diffusion. In: OJA, E.; KASKI, S. (Ed.). **Kohonen Maps**. Amsterdam: Elsevier, 1999. p.253–266.

GERS, F. A.; ECK, D.; SCHMIDHUBER, J. Applying LSTM to Time Series Predictable Through Time-Window Approaches. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL NEURAL NETWORKS, ICANN, 2001, Vienna, Austria. **Proceedings...** London: IEE, 2001.

GERS, F. A.; PéREZ-ORTIZAND, J. A.; ECK, D.; SCHMIDHUBER, J. Learning Context Sensitive Languages with LSTM Trained with Kalman Filters. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL NEURAL NETWORKS, ICANN, 2002. **Proceedings...** Berlin: Springer, 2002. p.655–660.

GOPPERT, J.; ROSENSTIEL, W. Neurons with continuous varying activation in self-organizing maps. In: INTERNATIONAL WORKSHOP ON ARTIFICIAL NEURAL NETWORKS. **From Natural to Artificial Neural Computation** Berlin: Springer-Verlag, 1995. p.419–426.

HAGENBUCHNER, M.; SPERDUTI, A. A Self-Organizing Map for Adaptive Processing of Structured Data. **IEEE Transactions on Neural Networks**, New York, v.14, n.3, p.491–505, May 2003.

HAYKIN, S. **Redes Neurais**: princípios e prática. 2.ed. São Paulo: Artmed, 1999.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. **Neural Computation**, Cambridge, v.9, n.8, p.1735–1780, 1997.

JAMES, D. L.; MIIKKULAINEN, R. SARDNET: a self-organizing feature map for sequences. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, 7, 1995, Cambridge, MA, USA. **Proceedings...** [S.l]: MIT Press, 1995. p.577–584.

JANG, J.-S. R. ANFIS: adaptive-network-based fuzzy inference systems. In: IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS, 1993. **Proceedings...** [S.l.: s.n.], 1993. v.23, n.3, p.665–685.

KOHONEN, T. The self-organizing map. **Proceedings of the IEEE**, [S.l.], v.78, n.9, p.1464–1480, Sept. 1990.

KOSKELA, T.; VARSTA, M.; HEIKKONEN, J.; KASKI, K. Recurrent SOM with local linear models in time series prediction. In: EUROPEAN SYMPOSIUM ON ARTIFICIAL NEURAL NETWORKS, ESANN, 6., 1998, Brussels, Belgium. **Proceedings...** [S.l.: s.n.], 1998. p.167–172.

LINåKER, F.; NIKLASSON, L. Time Series Segmentation Using an Adaptive Resource Allocating Vector Quantization Network Based on Change Detection. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS, IJCNN, 2000, Los Alamitos. **Proceedings...** [S.l.]: IEEE Computer Society, 2000. p.323–328.

MEAD, C. **Analog VLSI and Neural Systems**. Reading, MA, USA: Addison-Wesley, 1989. 371p.

MOZER, M. C. Neural net architectures for temporal sequences processing. In: WEIGEND, A. S.; GERSHENFELD, N. A. (Ed.). **Time series prediction**: forecasting the future and understanding the past. Reading, MA: Addison Wesley, 1993. v.15, p.243–264.

NILSSON, T. **KiKS is a Khepera Simulator**. 2001. Dissertação (Mestrado em Ciência da Computação) — Umeå University, Umeå (Sweden). Disponível em: <http://www.tstorm.se/projects/kiks/>. Acesso em: 20 jul. 2003.

NOLFI, S.; TANI, J. Extracting Regularities in Space and Time Through a Cascade of Prediction of Prediction Networks: the case of a mobile robot navigating in a structured environment. **Connection Science**, [S.l.], v.11, n.2, 1999.

SCHMIDHUBER, J. Adaptive history compression for learning to divide and conquer. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS, 1991, Singapore. **Proceedings...** [S.l.]: IEEE, 1991. v.2, p.1130–1135.

SCHMIDHUBER, J.; MOZER, M. C.; PRELINGER, D. Continuous History Compression. In: INTERNATIONAL WORKSHOP ON NEURAL NETWORKS, RWTH AACHEN, 1993. **Proceedings...** Augustinus: [s.n.], 1993. p.87–95.

STILES, B.; GHOSH, J. A habituation based neural network for spatio-temporal classification. In: IEEE WORKSHOP IN NEURAL NETWORKS FOR SIGNAL PROCESSING, 5, 1995, Cambridge, MA. **Proceedings...** [S.l.: s.n.], 1995. p.135–144.

STRICKERT, M.; HAMMER, B. Self-Organizing Context Learning. In: EUROPEAN SYMPOSIUM ON ARTIFICIAL NEURAL NETWORKS, ESANN, 2004. **Proceedings...** [S.l.: s.n.], 2004. p.39–44.

SUTTON, R. S. Learning to Predict by the Methods of Temporal Differences. **Machine Learning**, Dordrecht, v.3, p.9–44, 1988.

TANI, J. Learning to generate articulated behavior through the bottom-up and the top-down interaction processes. **Neural Networks**, [S.l.], v.16, n.1, p.11–23, 2003.

TANI, J.; NOLFI, S. Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor systems. **Neural Networks**, [S.l.], v.12, p.1131–1141, 1999.

TRUCCOLO, W. A.; DONG, D. W. Dynamic temporal decorrelation: information theoretic and biophysical model of the functional role of lateral geniculate nucleus. **Neurocomputing**, [S.l.], v.38-40, p.993–1001, 2001.

VARSTA, M.; HEIKKONEN, J.; R. MILLAN, J. del.**Context Learning with the Self-Organizing Map**. Finland: Helsinki University of Technology, 1997. Research Reports.

VOEGTLIN, T. Recursive self-organizing maps. **Neural Networks**, [S.l.], v.15, p.979–991, 2002.