Campus de São Carlos

"Redes neurais recorrentes para produção de
sequências temporais"

Autor:
Hélio D'Arbo Junior

Orientador:
Prof. Dr. Aluizio Fausto Ribeiro Araújo

UNIVERSIDADE DE SÃO PAULO

**ESCOLA DE ENGENHARIA
DE SÃO CARLOS**

REDES NEURAIS RECORRENTES PARA PRODUÇÃO DE SEQÜÊNCIAS TEMPORAIS

Hélio D'Arbo Junior

Dissertação apresentada à Escola de Engenharia de São Carlos, da Universidade de São Paulo, como parte dos requisitos para a obtenção do Título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Aluizio Fausto Ribeiro Araújo

São Carlos
1998

DEDALUS - Acervo - EESC

31100017208

Class.	TESE-EESC
Cod	1792
Tombo	70129/98

st 0980555

**Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca - EESC-USP**

D213r D'Arbo Junior, Hélio
Redes neurais recorrentes para produção de
seqüências temporais / Hélio D'Arbo Junior. --
São Carlos, 1998.

Dissertação (Mestrado) -- Escola de Engenharia
de São Carlos-Universidade de São Paulo, 1998.
Área: Engenharia Elétrica.
Orientador: Prof. Dr. Aluizio Fausto Ribeiro
Araújo.

1. Redes neurais recorrentes. 2. Planejamento.
3. Seqüências temporais. I. Título.

FOLHA DE APROVAÇÃO

Candidato: Engenheiro **HELIO D'ARBO JUNIOR**

Dissertação defendida e aprovada em 20-03-1998
pela Comissão Julgadora:

Prof. Doutor **ALUIZIO FAUSTO RIBEIRO ARAUJO (Orientador)**
(Escola de Engenharia de São Carlos - Universidade de São Paulo)

Prof. Doutor **ANDRÉ CARLOS PONCE DE LEON F. DE CARVALHO**
(Instituto de Ciências Matemáticas de São Carlos - Universidade de São Paulo)

Prof. Doutor **GERSON ZAVERUCHA**
(Universidade Federal do Rio de Janeiro - UFRJ)

Prof. Doutor **DENIS VINICIUS COURY**
Coordenador da Área de Engenharia Elétrica

JOSÉ CARLOS A. CINTRA
Presidente da Comissão de Pós-Graduação da EESC

**Aos meus pais,
Helio e Silvia,
por todo apoio e incentivo.**

AGRADECIMENTOS

Ao Prof. Dr. Aluizio Fausto Ribeiro Araújo, que mais que um orientador acadêmico, soube ser um grande amigo.

A Luciane Sempionato, pelo apoio, paciência e dedicação, sempre me incentivando durante esta caminhada, contribuindo muito para a conclusão deste trabalho.

A todas as pessoas que me ajudaram nesta caminhada, muitas delas caminharam comigo e hoje posso chamá-los de amigos. Em especial, aos companheiros Marcelo, Guilherme B., Arthur, Renato, João, Patrícia, Rodrigo D., Guilherme S., Rodrigo B..

Ao Departamento de Engenharia Elétrica pela utilização e fácil acesso aos seus laboratórios, bem como aos professores e funcionários que estiveram sempre dispostos a colaborar.

A Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo auxílio financeiro fornecido no período da realização deste trabalho, sem o qual o mesmo não seria possível.

SUMÁRIO

LISTA DE FIGURAS	i
LISTA DE TABELAS	viii
LISTA DE ABREVIATURAS E SIGLAS	x
LISTA DE SÍMBOLOS	xi
RESUMO	xiv
ABSTRACT	xv
1- INTRODUÇÃO	1
2- DESCRIÇÃO DO PROBLEMA	7
2.1- Sistema Discreto	9
2.2- Sistema Contínuo	13
2.2.1- PUMA 560	18
3- REVISÃO BIBLIOGRAFICA	20
3.1 - Redes Neurais	20
3.2 - Redes Neurais Recorrentes	22
3.2.1 - Transformação de sequência temporal em sequência espacial	24
3.2.2 - O <i>backpropagation</i> através do tempo e algumas de suas derivações..	26
3.2.3 - Arquiteturas LRGF.....	31
3.2.4 - Comparação entre desempenhos das arquiteturas	34
3.2.5 - Algoritmo de treinamento	36
3.3 - Redes neurais para lidar com planejamento e controle de trajetória de braço mecânico.....	40
4- MODELOS PROPOSTOS	46
4.1- Parte comum a todos os modelos	48
4.1.1- Algoritmo de aprendizado	50
4.2 - Modelo inicial	54

4.3 - Variação do modelo inicial	55
4.3.1- Incluindo realimentação local	55
4.3.2- Realimentação intercamada distribuída	57
4.4- Atuação dos modelos nos problemas discretos e contínuos	60
5- TESTE	62
5.1- Mundo dos blocos	64
5.1.1- Discussão dos resultados	72
5.2- Trajetória de um braço mecânico	73
5.2.1- Discussão dos resultados	87
6- CONCLUSÕES.....	88
7- REFERÊNCIAS BIBLIOGRÁFICAS	92
APÊNDICE	

LISTA DE FIGURAS

Figura 1: Seqüência de estados para movimentação de quatro blocos de uma posição inicial a uma posição meta. Os blocos estão representados pelas letras A, B, C e D e são desenhados por linhas cheias. Os quadrados pontilhados indicam possíveis posições para estes blocos.	9
Figura 2: Tipos de trajetórias consideradas para estudo. Elas têm em comum: (a) apenas o ponto meta, (b) apenas o ponto inicial, (c) nenhum ponto e (d) qualquer ponto dentro da seqüência. Nesta figura, cada círculo representa um estado inicial e cada “x” um estado final.	12
Figura 3: Braço mecânico PUMA 560.....	13
Figura 4: Problemas de cinemática direta e inversa.....	14
Figura 5: Tipos de juntas: (a) linear ou prismática, (b) rotacional, (c) de torção e (d) revolvente.	16
Figura 6: Parâmetros para os elos e juntas.....	17
Figura 7: Detalhes dos eixos de coordenadas para o PUMA 560 mostrado aqui com seus ângulos na posição zero.....	19
Figura 8: Modelo simplificado de neurônio utilizado em redes neurais artificiais.....	21
Figura 9: Diagramas do modelo BPTT: (a) Topologia do modelo; (b) Modelo desdobrado com alimentação direta para fins de treinamento.	26
Figura 10: Modelo de Elman. Neste modelo e em todos os outros de agora em diante, as linhas em negrito denotam conexões todos-para-todos e as linhas normais conexões um-para-um.	28
Figura 11: Modelo proposto por Jordan.....	30

Figura 12: Uma topologia geral para redes LRGF.....	32
Figura 13: Modelo proposto por Stornetta et al.....	32
Figura 14: Modelo proposto por Mozer.....	33
Figura 15: (a) Estruturas de modelos de processo sobre a hipótese erro de saída e (b) previsor neural associado.....	37
Figura 16: (a) Estruturas de modelos de processo sobre a hipótese NARMAX e (b) previsor neural associado.	38
Figura 17:(a) Estruturas de modelos de processo sobre a hipótese NARX, (b) previsor neural associado.	39
Figura 18: Robô manipulador bidimensional com dois graus de liberdade.....	40
Figura 19: (a) Código de endereços para cada posição espacial permitida para os blocos; (b) Uma configuração espacial particular dos blocos.....	47
Figura 20: Estrutura de alimentação direta comum a todos os modelos.....	49
Figura 21: Arquitetura de rede parcialmente recorrente proposta como referência....	55
Figura 22: Segundo modelo proposto no qual acrescenta-se realimentação intracamada na entrada.....	56
Figura 23: Terceiro modelo proposto no qual acrescenta-se realimentação intracamada na camada escondida.	57
Figura 24: Quarto modelo proposto: modelo referência no qual as conexões intercamadas são do tipo todos-para-todos.	57
Figura 25: Quinto modelo proposto: semelhante ao modelo 2. Transforma-se conexões de realimentação da camada de saída para a camada de entrada de um-para-um para todos-para-todos.	59

Figura 26: Sexto modelo proposto: semelhante ao modelo 3. Transforma-se conexões de realimentação da camada de saída para a camada de entrada de um-para-um para todos-para-todos.	60
Figura 27: Diagrama de blocos para o problema de planejamento de trajetória do mundo dos blocos.....	60
Figura 28: Diagrama de blocos para o problema de planejamento e controle de trajetória do braço mecânico.	61
Figura 29: Sequência de estados de uma trajetória do tipo $m-1$ criada para testar as arquiteturas.....	64
Figura 30: Exemplo de sequência de estados de trajetória do tipo 1-m utilizada no treinamento da rede.	65
Figura 31: Exemplo do tipo de trajetória m-m/sc corretamente recuperado.....	65
Figura 32: Exemplo do tipo de trajetória m-m/cc corretamente calculada.....	66
Figura 33: Um exemplo do tipo de trajetória m-m/cc corretamente recuperada	67
Figura 34: Número de épocas de treinamento por trajetória para cada arquitetura. ...	68
Figura 35: Percentagem de seqüências corretamente recuperadas quando a entrada da rede são o estado inicial e o estado meta treinado.....	69
Figura 36: Sequência de estados recuperados com falhas pela arquitetura 1 para a trajetória m-m/cc.	70
Figura 37: Percentagem de seqüências recuperadas para o teste de generalização.....	71
Figura 38: Trajetórias do tipo 1-m e m-1, criadas para o treinamento do braço mecânico.....	75
Figura 39: Trajetórias criadas para o tipo de trajetória m-m/sc. O ponto inicial é representado por I e o ponto meta é representado por M	75

Figura 40: Trajetórias do tipo m-m/cc criadas para treinar a rede. O ponto inicial é representado por I e o ponto meta é representado por M	76
Figura 41: Trajetória desejada e gerada pela arquitetura para o tipo m-1. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.	77
Figura 42: Ângulos do braço mecânico para as trajetórias do tipo m-1. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.	78
Figura 43: Torques desejados e gerados pela rede nos atuadores das juntas do braço mecânico para as trajetórias do tipo m-1. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.	79
Figura 44: Trajetórias obtidas pela rede e desejadas para o tipo 1-m. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.	80
Figura 45: Trajetórias geradas pela rede e desejadas para o tipo m-m/sc. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.	80
Figura 46: Trajetórias geradas pela rede e desejadas para o tipo m-m/cc. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.	81
Figura 47: Ângulos desejados e gerados pela rede, das juntas do manipulador para a trajetória tipo m-m/cc. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.	82
Figura 48: Torques gerados pela rede e desejados para os seis atuadores das juntas do manipulador para a trajetória tipo m-m/cc. Os pontos representados por	

círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.....83

Figura 49: Trajetórias obtidas pela rede e desejadas para o tipo 1-m com inserção de ruído de 5% para os pontos inicial e meta. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.....84

Figura 50: Trajetórias desejadas e geradas pela arquitetura para o tipo m-1 com inserção de ruído de 5% para os pontos inicial e meta. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.....85

Figura 51: Trajetórias desejadas e geradas pela arquitetura para o tipo m-m/cc com inserção de ruído de 5% para os pontos inicial e meta. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.....86

Figura 52: Trajetórias desejadas e geradas pela arquitetura para o tipo m-m/sc com inserção de ruído de 5% para os pontos inicial e meta. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.....86

FIGURAS DOS APÊNDICES

Figura 1: Ângulos desejados e gerados pela rede, das juntas do manipulador, para a trajetória tipo 1-m. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede..... IX

Figura 2: Torques desejados e gerados pela rede, das juntas do manipulador, para a trajetória tipo 1-m. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.....X

Figura 3: Ângulos desejados e gerados pela rede, das juntas do manipulador, para a trajetória tipo m-m/sc. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede..... XI

Figura 4: Torques desejados e gerados pela rede, das juntas do manipulador, para a trajetória tipo m-m/sc. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede..... XII

Figura 5: Ângulos desejados e gerados pela rede, das juntas do manipulador para a trajetória tipo l-m, com inserção de ruído de 5%. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede..... XIII

Figura 6: Torques desejados e gerados pela rede, das juntas do manipulador para a trajetória tipo l-m, com inserção de ruído de 5%. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.....XIV

Figura 7: Ângulos desejados e gerados pela rede, das juntas do manipulador para a trajetória tipo m-l, com inserção de ruído de 5%. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.....XV

Figura 8: Torques desejados e gerados pela rede, das juntas do manipulador para a trajetória tipo m-l, com inserção de ruído de 5%. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.....XVI

Figura 9: Ângulos desejados e gerados pela rede, das juntas do manipulador para a trajetória tipo m-m/cc, com inserção de ruído de 5%. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede..... XVII

Figura 10: Torques desejados e gerados pela rede, das juntas do manipulador para a trajetória tipo m-m/cc, com inserção de ruído de 5%. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede. XVIII

Figura 11: Ângulos desejados e gerados pela rede, das juntas do manipulador para a trajetória tipo m-m/sc, com inserção de ruído de 5%. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.XIX

Figura 12: Torques desejados e gerados pela rede, das juntas do manipulador para a trajetória tipo m-m/sc, com inserção de ruído de 5%. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.XX

LISTA DE TABELAS

Tabela 1: Constantes cinemáticas, α em graus, a e d estão em m.	II
Tabela 2: Valores da massa (kg) e centro de gravidade (m) dos elos.	II
Tabela 3: Momentos de inércia sobre centro de gravidade (kg-m^2).	II
Tabela 4: Inércia induzida (ARMATURE) e prazer suficiente	II
Tabela 5: Resultados obtidos no treinamento dos seis modelos, para o teste de produção de seqüências temporais, para a trajetória em que se tem vários pontos iniciais e um único ponto final.	IV
Tabela 6: Resultados obtidos no treinamento das seis arquiteturas, para o teste de produção de seqüências temporais, no caso em que se tem um único ponto inicial e vários pontos metas, sem pontos de cruzamento entre as trajetórias.....	IV
Tabela 7: Resultados obtidos no treinamento das seis arquiteturas, para o teste de produção de seqüências temporais, no caso em que se tem vários pontos iniciais e vários pontos metas, sem pontos de cruzamento entre as trajetórias.....	V
Tabela 8: Resultados obtidos no treinamento das seis arquiteturas, para o teste de produção de seqüências temporais, quando se tem vários pontos iniciais e vários pontos metas com cruzamento entre as trajetórias.	V
Tabela 9: Resultados obtidos no treinamento dos seis modelos, para o teste de generalização, para a trajetória em que se tem vários pontos iniciais e um único ponto final.	VI
Tabela 10: Resultados obtidos no treinamento das seis arquiteturas, para o teste de generalização, no caso em que se tem um único ponto inicial e vários pontos metas, sem pontos de cruzamento entre as trajetórias.....	VI

Tabela 11: Resultados obtidos no treinamento das seis arquiteturas, para o teste de generalização, no caso em que se tem vários pontos iniciais e vários pontos metas, sem pontos de cruzamento entre as trajetórias..... VI

Tabela 12: Resultados obtidos no treinamento das seis arquiteturas, para o teste de generalização, no caso em que se tem vários pontos iniciais e vários pontos metas, com pontos de cruzamento entre as trajetórias. VII

LISTA DE ABREVIATURA E SIGLAS

PUMA	- Programmable Universal Machine Assembly
DC	- Direct Current
LGRF	- Locally Recurrent Globally Feedforward
BPTT	- Backpropagation through time
BBPTT	- Batch backpropagation through time
QPTT	- Quickprop through time
RTRL	- Real time recurrent learning
MLP	- Multilayer perceptron
NARMAX	- Non-linear Auto-Regressive Moving Average with Exogeneous Inputs
NARX	- Non-linear Auto-Regressive with Exogeneous Inputs
m-1	- trajetória onde se tem vários pontos iniciais e um único ponto meta
1-m	- trajetória onde se tem apenas um ponto inicial e vários pontos metas
m-m/sc	- trajetória em que se tem vários pontos iniciais e vários pontos metas onde as trajetórias não têm pontos em comum
(m-m/cc)	- trajetória onde se tem vários pontos iniciais e vários pontos finais com ponto de cruzamento.

LISTA DE SÍMBOLOS

$g(b_1, b_2, \dots, b_n)$	- função com valores reais contínuos.
$p(v)$	- função contínua da variável v .
$\phi_{ij}(\cdot)$	- função contínua monotonicamente crescente.
θ_i	- ângulo entre os eixos x_{i-1} e x_i sobre o eixo z_{i-1} .
d_i	- distância da origem do $(i-1)$ -ésimo sistema de coordenada para a interseção do eixo z_{i-1} com o eixo x_i ao longo do eixo z_{i-1} . Se o eixo z_{i-1} e o eixo x_i não se cruzarem então é a distância perpendicular entre os eixos x_i e x_{i-1} .
α_i	- distância da origem do i -ésimo sistema de coordenada para a interseção dos eixos z_{i-1} e x_i ao longo do eixo x_i .
α_i	- ângulo de rotação medido do eixo z_{i-1} positivo, ou sua projeção paralela, para o eixo z_i positivo sobre o eixo x_i positivo, onde a direção positiva é o sentido horário.
x	- posição espacial referente ao eixo x do sistema de coordenada cartesiana.
y	- posição espacial referente ao eixo y do sistema de coordenada cartesiana.
z	- posição espacial referente ao eixo z do sistema de coordenada cartesiana.
ω_{ij}	- pesos sinápticos.
t	- variável que representa tempo.
$y(t)$	- saída do neurônio.
$f(\cdot)$	- qualquer função de ativação linear ou não linear.
ω_o	- limiar.
$i_i(t)$	- ativação da i -ésima unidade da camada de entrada.
$h_i(t)$	- ativação da i -ésima unidade da camada escondida.
$c_i(t)$	- ativação da i -ésima unidade da camada de contexto.
$o_i(t)$	- ativação da i -ésima unidade da camada de saída.
T	- número de passos de tempo da sequência.

n_i	- número de unidades da camada de entrada.
n_e	- número de unidades da camada escondida.
n_c	- número de unidades da camada de contexto.
n_s	- número de unidades da camada de saída.
$\lambda_j(t)$	- estado de ativação de uma unidade de processamento.
β	- força das conexões de realimentação.
a	- constante diferente de zero.
b	- constante diferente de zero
$x_j(t)$	- conjunto de entradas das unidades de processamento.
$G_j(z^{-1})$	- é uma função de transferência linear.
n_p	- número de polos.
n_z	- número de zeros.
m	- número inteiro.
$q(t)$	- ativação da unidade de processamento, antes da função de ativação.
$s_i(t)$	- estímulos fornecidos pelo ambiente.
L_i	- comprimento dos elos das juntas .
A	- matriz Jacobiana.
$\theta_i(t)$	- posição angular de cada junta “i”.
$\dot{\theta}_i(t)$	- velocidade angular de cada junta “i”.
$\ddot{\theta}_i(t)$	- aceleração angular de cada junta “i”.
$\tau_i(t)$	- torques de cada junta “i”.
$\vec{r}$	- evento qualquer.
$\vec{v}$	- resultado que ocorre no contexto do evento $\vec{x}$.
E	- erro médio quadrático.
D_i^p	- saída desejada para a unidade i da camada de saída da rede quando apresentado o padrão p .
S_i^p	- saída obtida para a unidade i da camada de saída da rede quando apresentado o padrão p .
δ_i	- delta calculado para o algoritmo de aprendizagem de retropropagação.

- η - taxa de aprendizagem.
- ψ - termo de *momentum*.
- is_i - estado inicial apresentado à i-ésima unidade de entrada.
- gs_i - estado final apresentado à i-ésima unidade de entrada.
- k_{ij} - pesos sinápticos.
- $w(t)$ - sequência de ruído branco.
- $U_p(t-1) = \{u_p(t-1), u_p(t-2), \dots, u_p(t-N)\}$ - vetor com as últimas N variáveis de estado.
- $X(t-1) = \{x(t-1), x(t-2), \dots, x(t-M)\}$ - vetor com os M últimos valores de entrada.
- $W(t-1) = \{w(t-1), w(t-2), \dots, w(t-P)\}$ - é um vetor com os P últimos valores para o ruído.

RESUMO

D'ARBO JR., H., *Redes Neurais recorrentes para produção de seqüências temporais*. São Carlos, 1998. 95p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo.

Dois problemas de planejamento de trajetórias são tratados nesta dissertação, sendo um discreto e outro contínuo. O problema discreto consiste em estabelecer todos os estados intermediários de uma trajetória para levar um conjunto de quatro blocos de uma posição inicial à uma posição meta. O problema contínuo consiste em planejar e controlar a trajetória do braço mecânico PUMA 560. A classe de modelos que se utilizou nesta dissertação foram os modelos parcialmente recorrentes. O problema discreto foi utilizado com a finalidade de comparar os seis modelos propostos, buscando obter um modelo com bom desempenho para resolução de problemas de produção de seqüências temporais. Para o problema contínuo aplicou-se apenas o modelo que apresentou melhor desempenho na resolução do problema discreto. Em ambos os casos são apresentados como entrada para a rede, o ponto inicial e o ponto meta. Dois tipos de testes foram aplicados as arquiteturas: teste de produção e de generalização de seqüências temporais. Para cada problema foram criados quatro tipos distintos de trajetórias, com graus de complexidades diferentes. Para o problema discreto, em média, a arquitetura com realimentação da camada de saída para a camada de entrada e da camada de entrada para ela mesma, todos-para-todos, foi a que apresentou menor número de épocas e também os menores valores de erro durante o treinamento. Foi o único que conseguiu recuperar todos os padrões treinados e de forma geral apresentou melhor capacidade de generalização. Por isto, este modelo foi escolhido para ser aplicado na resolução do problema contínuo, tendo bom desempenho, conseguindo reproduzir as trajetórias treinadas com grande precisão. Para o problema discreto todos os modelos apresentaram baixa capacidade de generalização. Para o problema contínuo o modelo abordado apresentou-se de forma satisfatória mediante o acréscimo de ruído.

Palavras-Chave: Redes neurais recorrentes, planejamento, seqüências temporais,

ABSTRACT

D'ARBO JR., H. (1998), *Recurrent neural networks for Production of temporal sequences*. São Carlos, 1998. 95p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo.

Two trajectory planning problems are discussed in this work, one of them being discrete and the other continuous. The discrete problem consists in establishing all the intermediate states of a trajectory to move a set of four blocks from a initial to a goal position. The continuous problem consists in planning and controlling the trajectory of the PUMA 560 mechanical arm. The class of models utilized in this work were the partially recurrent models. The discrete problem was used in order to compare the six proposed models, aiming at the acquisition of a model with a good performance for the resolution of production of temporal sequence problems. For the continuous problem, only the model that presented better performance in solving the discrete problem was applied. The initial and goal point are presented as input for the network in both problems. Two types of tests were applied to the architectures: production and generalization of temporal sequence tests. Four distinct types of trajectories with different complexity levels were created for each problem. In average, for the discrete problem, the architecture with feedback from the output to the input layer and from input layer to itself all-to-all presented the lowest epoch number in addition to the lowest error values during the training. This was the only model that managed to recover all the patterns trained and in general presented better generalization capacity. For this reason, this model was chosen to be applied in the resolution of the continuous problem. It presented a good performance to the production of mechanical arm trajectories, managing to reproduce the trained trajectories with great accuracy. For the discrete problem, all the models presented low generalization capacity. For the continuous problem, the approached model presented itself in a satisfactory manner by means of noise addition.

Key words: Recurrent neural networks, planning, temporal sequence.

1. INTRODUÇÃO

A arquitetura perceptron multicamadas padrão é conhecida como um aproximador universal de funções [14]. Mapeamentos estáticos realizados por redes neurais com alimentação direta (*feedforward*) têm sido frequentemente aplicados em predição de sistemas dinâmicos. Entretanto, para modelar tais sistemas dinâmicos, pode ser mais apropriado usar redes neurais recorrentes. Em estruturas recorrentes, considera-se também os estados internos da rede em adição ao mapeamento estático entre entrada e saída. Estes estados funcionam como uma memória de curto prazo e são capazes de representar informações sobre os estados anteriores da rede.

Redes completamente recorrentes são redes em que todas as unidades de processamento são interconectadas com todas as outras unidades desta arquitetura. Redes parcialmente recorrentes são caracterizadas por suas estruturas serem predominantemente de alimentação direta, mas por possuírem conexões de realimentação cuidadosamente escolhidas. Modelos destas duas arquiteturas têm sido propostos para lidar com tarefas que envolvam seqüências temporais [17]. Entretanto, ainda não é claro qual algoritmo e arquitetura são mais apropriados para determinados tipos de problemas. Isto motiva o autor deste documento a propor e estudar redes neurais recorrentes para uma dada tarefa.

Levado pela idéia de que estados internos podem representar o passado em processos iterativos, propõe-se nesta dissertação seis modelos de redes neurais parcialmente recorrentes, buscando estudar o papel desempenhado pela realimentação entre unidades de processamento da mesma camada e unidades de processamento de camadas diferentes na resolução de problemas que lidam com seqüências temporais. Quatro são os tipos de problemas que lidam com seqüências temporais [17]: reconhecimento de seqüências temporais, reprodução de seqüências temporais, produção de seqüências temporais e associação de seqüências temporais.

Nesta dissertação, será dada ênfase a problemas de produção de seqüências temporais em que são apresentados os pontos extremos de uma dada seqüência temporal para que a rede gere toda esta seqüência.

Todas as arquiteturas propostas terão sua parte de alimentação direta escolhidas com base no teorema de Kolmogorov [27]. Este teorema estabelece que qualquer função com valores reais contínuos $g(b_1, b_2, \dots, b_n)$ definida em $[0,1]^n$, $n \geq 2$, pode ser representada na forma:

$$g(b_1, b_2, \dots, b_n) = \sum_{j=1}^{2n+1} p_j \left[\sum_{i=1}^n \phi_{ij}(b_i) \right] \quad (1.1)$$

onde os termos p_j são funções contínuas de uma variável, apropriadamente escolhidas, e as funções ϕ_{ij} são funções contínuas monotonicamente crescentes independentes de g .

Este teorema sugere que uma rede com alimentação direta possuindo uma camada escondida com um número suficiente de unidades e tendo a função de ativação do tipo sigmoidal é capaz de representar qualquer função contínua para qualquer precisão desejada [15], [16], [31], [50]. Com base neste teorema, os seis modelos de redes neurais parcialmente recorrentes propostos nesta dissertação, que se caracterizam por possuírem uma mesma estrutura de alimentação direta, apresentam esta estrutura de alimentação direta basicamente uma arquitetura perceptron multicamadas com 3 camadas: uma camada de entrada, uma camada escondida e uma camada de saída.

Inicialmente um modelo recorrente é proposto como referência. Este modelo consiste basicamente na estrutura de alimentação direta com conexões de realimentação do tipo um-para-um da camada de saída para a camada de entrada. Os outros cinco modelos são variações deste, acrescentando-se realimentação local na camada de entrada e na camada escondida. Outra variação no modelo padrão é quanto ao tipo de conexão de realimentação entre as unidades de processamento: um-para-um ou todos-para-todos.

Os modelos estudados são basicamente arquiteturas multicamadas com alimentação direta. Neste caso, as realimentações podem ser consideradas, para fins de treinamento, como uma extensão da camada anterior àquela que recebe a

realimentação, viabilizando o emprego do algoritmo de retropropagação (*backpropagation*) [45]. Este algoritmo é o mais empregado e o mais estudado em redes neurais. Apresenta uma capacidade de armazenamento consideravelmente grande e é relativamente de simples implementação. A introdução do algoritmo de retropropagação tem um papel muito importante no ressurgimento dos interesses em redes neurais artificiais e tem expandido muito a quantidade de problemas aos quais as redes neurais podem ser aplicadas.

O problema tratado nesta dissertação consiste em planejamento e controle de trajetória. O problema de planejamento e controle de trajetória de sistemas não-lineares é de grande importância e apresenta um grau de complexidade consideravelmente alto dentro da área de controle. Quando se trata de problemas em que o sistema deve atuar em um ambiente não-estruturado e sujeito a ruídos, o grau de complexidade torna-se ainda maior.

Um problema básico em robótica é o planejamento de movimentos para executar uma tarefa pré-especificada e controlar o robô para que ele realize os comandos necessários para executar estas ações. Para planejar e controlar um movimento voluntário, o cérebro deve determinar uma única trajetória entre um número infinito de possíveis trajetórias e deve ativar músculos para realizar a trajetória desejada.

O controle de um manipulador robô para executar a tarefa de coletar um objeto em um espaço de trabalho e levá-lo a uma posição desejada pode ser resolvido de forma precisa utilizando os métodos de robótica clássica [29]. Em geral, os sistemas de controle de um braço mecânico necessitam de grande precisão. Quando estes métodos 'tradicionais' são usados, é necessário que seja feita uma calibragem do sistema para que ele possa operar com a precisão requerida. Entretanto, estes métodos mostram-se ineficientes em sistemas que sofrem desgastes, ou seja, caso ocorram mudanças na posição dos sensores ou na posição da base do manipulador com relação ao sistema de coordenadas tomado como base para o equacionamento do manipulador, pois são necessárias freqüentes recalibrações. Tais métodos também apresentam fraco desempenho em tarefas que lidam com controle de sistemas complexos e não-lineares atuando em condições de incerteza e/ou de ruído. Algumas

destas limitações podem ser resolvidas utilizando sistemas de controle com capacidade de auto-adaptação. Tais estratégias de controle precisam ter capacidade de se adaptar ao meio e aprender de maneira autônoma.

Para este tipo de problema, a área de Inteligência Artificial clássica parece ser promissora. No entanto, a aplicação de tais técnicas em robótica também apresenta limitações significativas, destacando-se as dificuldades em gerar e armazenar todas as possibilidades de situações suficientemente precisas do robô e do ambiente. Assim sendo, as estratégias de controle devem se adaptar a novas situações a partir do conhecimento parcial de seu universo de atuação. Nesta linha, o paradigma rede neural artificial é adequado para treinar os controladores do robô através de demonstração e não por programação de tipos de comportamento de aprendizagem, permitindo assim, projetar sistemas robóticos mais flexíveis.

O problema de planejamento e controle de trajetórias consiste basicamente em produção de seqüências temporais. Para o modelamento de tais sistemas dinâmicos, utiliza-se nesta dissertação redes neurais parcialmente recorrentes. Os modelos apresentam realimentação intercamada da camada de saída para a camada de entrada e realimentação intracamada localizada na camada de entrada ou na camada escondida. Espera-se que as realimentações intracamadas permitam aos modelos levar em conta características do passado para lidar com situações ambíguas no presente e que as realimentações intercamadas forneçam aos modelos informações do passado mais recente para definir o estado presente. A combinação destes dois tipos de realimentação deve dar à rede a capacidade de produzir seqüências temporais.

A fim de analisar o desempenho dos modelos propostos na resolução de problemas de planejamento de trajetória, com o intuito de escolher o melhor modelo para ser utilizado no planejamento de trajetórias contínuas, propõe-se inicialmente um problema de planejamento de trajetória discreto, pois este apresenta estados bem definidos e número finito de estados, o que torna mais fácil observar e mensurar os resultados obtidos. O problema discreto consiste em estabelecer uma seqüência de estados que leva um conjunto de quatro blocos de uma posição espacial inicial até uma posição espacial final. Um bloco pode ser colocado em qualquer uma de três

colunas com quatro posições possíveis cada. Para o treinamento das arquiteturas foram criados quatro tipos distintos de trajetórias, cada uma apresentando um grau de complexidade diferente. Como entrada para os modelos são apresentados os pontos inicial e final da trajetória. Como saída, cada modelo deve produzir todos os pontos que vão desde o estado inicial até o estado final.

O problema discreto tem a mesma natureza do problema contínuo: dados os estados iniciais e finais de uma trajetória, o modelo de rede neural deve estabelecer todos os estados intermediários da trajetória. Contudo, o problema contínuo apresenta algumas características que aumentam sua complexidade quando comparado com o problema discreto, isto é, seus estados não são bem definidos, logo, podem apresentar estados aproximados aos estados determinados e também podem apresentar um número infinito de estados. O problema contínuo que testa o modelo proposto consiste em planejar e controlar a trajetória de um braço mecânico. A rede terá como entrada as posições espaciais x , y e z , os ângulos e os torques em cada junta do manipulador para as posições inicial e final desejadas. A rede produzirá uma sequência de estados das variáveis mencionadas que levam o manipulador da posição inicial à sua meta.

Os testes aplicados às arquiteturas foram:

- Teste de produção de sequências temporais onde se verifica a capacidade da rede de recuperar sequências treinadas.
- Teste de generalização onde se verifica a capacidade das arquiteturas de produzir sequências não-treinadas.

O Capítulo 2 apresenta uma descrição detalhada dos problemas discreto e contínuo que serão tratados nesta dissertação, buscando justificar o uso de redes neurais recorrentes e destacar as características e dificuldades na resolução de tais problemas. No Capítulo 3, apresenta-se a revisão bibliográfica, trazendo definições e conceitos de redes neurais e redes neurais recorrentes para melhor situar o leitor. Também neste capítulo, destaca-se alguns modelos de redes neurais recorrentes propostos por outros autores e também alguns exemplos de redes neurais aplicadas

ao problema de planejamento e controle de trajetória de braço mecânico. A finalidade maior deste capítulo é apresentar os trabalhos que contribuíram para esta pesquisa destacando e justificando características desejadas para os modelos propostos. No Capítulo 4, apresenta-se os modelos propostos para a resolução dos problemas destacando suas características e justificando a escolha destes. O Capítulo 5 apresenta os testes, os resultados e a discussão destes resultados para ambos os problemas. O Capítulo 6 apresenta as conclusões para o trabalho.

2. DESCRIÇÃO DO PROBLEMA

Redes neurais com alimentação direta são aplicadas com grande eficiência para modelar sistemas estáticos, ou seja, são treinadas para produzir um padrão de saída em resposta a um padrão particular de entrada. Entretanto, em muitas aplicações científicas e de engenharia é necessário modelar processos dinâmicos que envolvem seqüências temporais. Hertz *et al.* [17] definem três classes de problemas que lidam com seqüências temporais:

- Reconhecimento de seqüências: Aqui se deseja gerar um padrão de saída particular quando uma seqüência de entrada específica é apresentada. Não é necessário reproduzir a seqüência de entrada.
- Reprodução de seqüência: Neste caso, a rede deve ser capaz de gerar o resto de uma seqüência quando parte dela é vista, isto é, a rede deve apresentar toda a seqüência. Isto pode ser entendido como uma generalização da auto-associação.
- Associação temporal: Neste caso geral, uma seqüência de saída particular deve ser gerada em resposta a uma seqüência específica de entrada. As seqüências de entrada e de saída devem ser diferentes, portanto ocorre generalização da heteroassociação para padrões dinâmicos. Isto inclui como caso especial a geração de seqüências e os dois casos anteriores.

Esta dissertação trata de um caso mais geral de reprodução de seqüências temporais que será chamado de produção de seqüências temporais. Nesta situação, a rede deve ser capaz de gerar toda uma seqüência quando seus pontos inicial e final

são dados e deve ser capaz de gerar novas seqüências não-treinadas a partir daquelas conhecidas.

Nesta dissertação, trata-se de dois problemas de produção de seqüências temporais. Um consiste em planejamento de trajetória em sistemas discretos, o outro, em planejamento e controle de trajetória em sistemas contínuos. Em ambos os problemas, o objetivo é sair de uma posição inicial para uma posição meta passando por pontos intermediários que caracterizam a trajetória desejada. A rede deve portanto, ser capaz de gerar uma seqüência temporal de estados intermediários para ir do ponto inicial ao ponto meta em uma quantidade limitada de ciclos. As seqüências a serem geradas apresentam tamanhos variados.

Será verificada a capacidade dos modelos em duas situações distintas:

- A capacidade da rede para produzir uma seqüência de estados intermediários entre o ponto inicial e o ponto meta sendo, estes treinados.
- A capacidade da rede para generalizar, ou seja, gerar uma seqüência de pontos entre o ponto inicial e o ponto meta, sendo que estes não tenham sido treinados.

Esta família de tarefas requer uma rede neural recorrente com conexões assimétricas. Tanto uma rede de alimentação direta como uma rede com conexões simétricas não seriam capazes de realizar este tipo de tarefa pois estas necessariamente convergem para um estado estacionário [17].

Neste trabalho, diferentes modelos de redes neurais parcialmente recorrentes serão aplicados para resolver o problema de produção de seqüências temporais em um sistema discreto, com o objetivo de avaliar o desempenho destes modelos, destacando características favoráveis e desfavoráveis para a resolução de tais tipos de problemas. O modelo que apresentar melhor resultado será aplicado na resolução do problema de produção de seqüências temporais em sistema contínuo. Com isto, busca-se um modelo geral que funcione bem tanto para sistemas discretos quanto para sistemas contínuos. A seguir, serão descritos detalhes do problema discreto e do problema contínuo tratados nesta dissertação.

2.1- Sistema Discreto

Os modelos propostos são aplicados inicialmente na reprodução de seqüências temporais em sistemas discretos. Neste caso, o problema consiste em estabelecer uma seqüência de estados que levem um conjunto de quatro blocos de uma posição espacial inicial para uma posição final sobre uma mesa, como o exemplo da Figura 1. Cada bloco pode ocupar uma das doze posições estabelecidas por três colunas com quatro posições permitidas cada. Os movimentos têm as seguintes restrições:

- Cada bloco pode ocupar qualquer posição vazia em qualquer uma das três colunas desde que a posição imediatamente abaixo esteja ocupada por outro bloco, ou pela mesa.
- As posições iniciais e finais podem ser qualquer uma dentre as permitidas.
- É permitido mover apenas um bloco por vez.
- Um bloco pode ser transferido somente se seu topo estiver livre.
- Um bloco pode ser colocado sobre o topo de outro bloco livre ou sobre a mesa.
- Um bloco pode ser deslocado por mais que uma coluna em um único movimento.

Figura 1: Seqüência de estados para movimentação de quatro blocos de uma posição inicial a uma posição meta. Os blocos estão representados pelas letras A, B, C e D e são desenhados por linhas cheias. Os quadrados pontilhados indicam possíveis posições para estes blocos.

No domínio da Inteligência Artificial, McDermott e Hendler [37] afirmam que “... planejamento é projetar o comportamento de alguma entidade que atua: um indivíduo, um grupo ou uma organização. A saída é um tipo de esquema para o comportamento, o qual é chamado de plano”.

Recentemente, McAllester e Rosenblitt [35] propuseram o algoritmo SNLP para resolver este problema. Este algoritmo produz uma sequência totalmente ordenada de ações, resultante da busca em um espaço de planos parciais. O método produz satisfatoriamente um plano completo estendendo um plano parcial. O resultado da busca é uma sequência de operadores que podem transformar um conjunto de assertivas que definem o estado inicial em outro que esboça o estado final. Este elegante algoritmo apresenta limitações como: todas as possíveis saídas deste plano necessitam ser conhecidas a priori e há a possibilidade de explosão combinatória durante a busca.

Uma das alternativas atuais para lidar com intratabilidade de planejamento é usar soluções de problemas de planejamento similares para produzir novos planos. Esta abordagem é chamada de Planejamento Baseado em Casos ou Planejamento Baseado em Memória [37], [10]. Propõe-se nesta dissertação um modelo de rede neural para realizar Planejamento Baseado em Casos. Assim, o objetivo do modelo a ser proposto é gerar um conjunto completo de situações partindo do estado inicial e chegando ao estado final desejado. O modelo realizara planejamento baseado em casos, visto que aprende de casos, ou seja, utiliza informações de estados anteriores para novos estados e também cada sequência aprendida leva em conta a anterior. A trajetória resultante é baseada nas trajetórias aprendidas pela rede. Isto é, dadas as disposições inicial e final dos blocos, a rede deve gerar os estados intermediários para levar os blocos de uma situação à outra. As seguintes suposições são assumidas a fim de usar uma rede neural recorrente para realizar planejamento baseado em casos:

- As restrições do planejamento são absorvidas durante o estágio de aprendizagem da rede;

- Os operadores emergem da dinâmica da rede que produz as operações permitidas como função dos padrões apresentados, do histórico da rede, da meta e das restrições implicitamente absorvidas;
- Durante a fase de teste, as entradas da rede são somente os estados inicial e final de cada sequência.

Idealmente, o planejador neural deve generalizar, isto é, a sequência deve ser recuperada quando inicializada a partir de um estado da sequência que seja diferente daquele que foi treinado anteriormente como estado inicial. Embora lide com planejamento baseado em casos, está fora do escopo desta dissertação comparar este algoritmo com outros que realizem planejamento baseado em casos.

Com a finalidade de avaliar o desempenho dos modelos, foram estabelecidas quatro classes de trajetórias que diferem entre si pelo tipo de interseção entre elas. Cada trajetória apresenta um grau de complexidade diferente baseado no número de cruzamentos entre ela e as outras trajetórias e também no número de pontos iniciais e finais diferentes das trajetórias que serão armazenadas. Assim, o planejador neural deverá ser capaz de lidar com: trajetórias que iniciam em pontos diferentes e convergem para uma única meta (Figura 2.a); trajetórias que iniciam em um mesmo ponto e terminam em diferentes metas (Figura 2.b); trajetórias que partem de diferentes pontos iniciais e chegam a diferentes metas trajetórias sem pontos em comum (Figura 2.c) e com pontos intermediários de cruzamento (Figura 2.d).

A trajetória apresentada na Figura 2.a pode ser considerada a de menor grau de complexidade. Uma vez que se deseja sair de vários pontos iniciais e chegar a um único ponto final, este último pode se tornar um forte atrator, o que pode ser um ponto positivo pois facilitaria à rede aprender a alcançar o ponto meta. Por outro lado, um atrator forte poderia ter consequências indesejadas como dificultar à rede aprender pontos intermediários entre o inicial e o desejado pois durante a fase de aprendizagem das sequências, o mesmo ponto meta aparece mais do que os outros pontos a serem treinados. O segundo tipo de trajetória, Figura 2.b, apresenta um grau de complexidade maior em relação ao anterior, pois se deseja sair de um mesmo ponto e chegar a vários pontos distintos. Neste caso, o ponto meta não é mais

considerado um ponto atrator forte, aumentando portanto a complexidade do problema. O terceiro tipo de trajetória, Figura 2.c, apresenta um grau de complexidade maior do que os dois anteriores, pois se deseja partir de pontos distintos no espaço e chegar a pontos também distintos no espaço, não possuindo pontos de cruzamento entre as trajetórias. Neste tipo de trajetória não existe nenhum atrator bem definido. Dependendo da disposição dos pontos iniciais e finais das diferentes trajetórias no espaço, a rede pode encontrar dificuldade em encontrar uma função para mapear o espaço de trajetória. O último tipo de trajetória abordada, Figura 2.d, é a que apresenta maior grau de complexidade. Para este tipo de trajetória, assim como no tipo anterior, deseja-se partir de pontos distintos no espaço e chegar a pontos também distintos no espaço, mas agora ocorrem pontos de cruzamento entre as trajetórias.

O papel do contexto em cada uma das quatro trajetórias é de fundamental importância, pois permite gerar o próximo ponto da sequência sem ter uma nova entrada externa. Para as trajetórias onde há pontos de cruzamento, Figura 2d, o contexto tem uma influência ainda maior, pois quando se chega a um ponto de cruzamento entre duas ou mais trajetórias, faz-se necessário levar em conta o passado, ou seja, saber de onde se vem, para se optar pela trajetória correta até o ponto meta.

Figura 2: Tipos de trajetórias consideradas para estudo. Elas têm em comum: (a) apenas o ponto meta, (b) apenas o ponto inicial, (c) nenhum ponto e (d) qualquer ponto dentro da sequência. Nesta figura, cada círculo representa um estado inicial e cada “x” um estado final.

Embora o sistema possua entradas e saídas discretas, todo o processamento interno das arquiteturas é tratado como contínuo, ou seja, a função de transferência da camada escondida e da camada de saída é a função sigmoideal, que é uma função contínua. Na camada de saída, após a função de transferência, o sinal passa por uma

função degrau que torna a saída discreta. Não se usou a função degrau diretamente como a função de transferência das unidades de processamento, pois a rede se mostrou mais instável, ou seja, demora mais para convergir. Os modelos abordados tratam o tempo implicitamente. As propriedades consideradas para a avaliação de desempenho dos modelos são: papel de recorrência, capacidade de produção de seqüência dados os pontos inicial e meta e capacidade de generalização.

2.2- Sistema Contínuo

Em um segundo momento, aplicou-se a arquitetura que apresentou melhor desempenho na resolução do problema anterior para resolver um problema de produção de seqüência temporal em sistema contínuo. O problema consiste em planejar e controlar a trajetória de um braço mecânico em um ambiente não-estruturado e sujeito a ruído. O braço mecânico utilizado foi o PUMA 560, Figura 3, que é um dos robôs mais comuns em laboratórios de universidades e também um dos mais comuns em linha de montagem.

Figura 3: Braço mecânico PUMA 560.

Para o controle de movimentos voluntários, três tarefas devem ser resolvidas: planejamento de trajetória, cinemática inversa e dinâmica inversa. Um problema é bem proposto quando sua solução existe, é única e depende continuamente dos dados iniciais. Um problema mal proposto falha em satisfazer um ou mais destes critérios [23]. Tais tarefas são consideradas mal propostas no sentido em que a solução destas não é única. O problema de planejamento de trajetória não é único visto que, quando os pontos inicial e final são especificados, existe um número infinito de trajetórias que satisfaz estas condições. O problema de cinemática inversa não possui uma única solução quando o manipulador for redundante, ou seja, possui um número maior de graus de liberdade do que o necessário para se executar a tarefa, havendo para uma mesma posição espacial várias combinações de ângulos das juntas. O problema da dinâmica inversa também não possui uma solução única, pois mesmo quando o período de tempo para o ângulo da junta é especificado, há um número infinito de formas de onda para o torque no atuador da junta, que alcança o ângulo desejado no mesmo período de tempo.

Cinemática é o estudo do movimento de corpos rígidos sem considerar as massas e forças que causam o movimento [7], [13], [26]. A cinemática de manipuladores lida com o estudo analítico da geometria do movimento do braço do robô com relação a um sistema de coordenadas fixas tomado como referência, em função do tempo. Isto é, a cinemática estuda como os vários elos se movem com respeito a um outro em um período de tempo.

Figura 4: Problemas de cinemática direta e inversa.

O problema de cinemática pode ser dividido em cinemática direta e cinemática inversa. O subproblema de cinemática direta consiste em obter a posição e a orientação do efetuador com respeito a um sistema de coordenadas tomado como referência, dados os ângulos das juntas do manipulador e os parâmetros geométricos dos elos. Para o subproblema de cinemática inversa são dadas: posição e orientação desejadas para o efetuador e os parâmetros geométricos dos elos com relação a um sistema de coordenadas tomado como referência e deseja-se obter a combinação de ângulos das juntas para a especificada posição. Um simples diagrama de blocos indicando a relação entre estes dois subproblemas é mostrado na Figura 4.

O problema da dinâmica dos manipuladores está relacionado com as equações de movimento, o caminho em que o manipulador se move em resposta a torques aplicados pelos atuadores ou forças externas [7], [13], [26]. O problema da dinâmica de manipuladores também envolve dois subproblemas importantes: dinâmica direta e dinâmica inversa. O problema da dinâmica direta do braço mecânico visa descrever o comportamento dinâmico, dadas as forças e os momentos que regem seus movimentos em função do tempo. O problema de dinâmica inversa objetiva transformar ângulos e comprimento das juntas em ações de controle. Consiste em calcular os torques nas juntas, requeridos para um dado conjunto de ângulos, velocidade e aceleração destas juntas.

O problema de planejamento de trajetória estabelece um caminho formado por um conjunto de pontos que o sistema deve seguir para sair de uma posição de origem e chegar à posição meta [7], [13], [26]. O caminho pode ser especificado em coordenadas das juntas ou em coordenadas cartesianas. Esta última é a maneira mais usual de especificar a posição, a orientação, a velocidade e a aceleração do braço.

Robôs típicos são manipuladores constituídos de elos conectados serialmente por juntas. Cada elo é movido por um atuador. O movimento dos elos é uma rotação angular ou uma translação retilínea. Cada junta tem um grau de liberdade podendo ser de quatro tipos: linear ou prismática (Figura 5.a), rotacional (Figura 5.b), de torção (Figura 5.c) e revolvente (Figura 5.d). Para manipuladores com n juntas numeradas de 1 a n , há $n+1$ elos numerados de 0 a n . O elo 0 é a base do

manipulador, geralmente fixo, e o elo n carrega o efetuador. A junta i conecta o elo i ao elo $i-1$.

Figura 5: Tipos de juntas: (a) linear ou prismática, (b) rotacional, (c) de torção e (d) revolvante.

Para a análise de um sistema manipulador, é uma prática comum definir um sistema de coordenadas globais que é geralmente fixo à base do manipulador e é tomado como referência e um sistema de coordenadas locais fixas em cada elo. Este último se move com o elo. Denavit e Hartenberg [8] propuseram um método sistemático e generalizado utilizando matriz algébrica para descrever a geometria espacial dos elos de um manipulador com relação ao sistema de coordenadas globais; a representação D-H. Esta, define um sistema de coordenadas locais para cada elo.

A representação D-H de um elo rígido depende de quatro parâmetros geométricos associados com cada elo. Estes quatro parâmetros descrevem completamente qualquer junta de revolução ou prismática. Os parâmetros para os elos e juntas (Figura 6) podem ser resumidos como segue:

- θ_i : é o ângulo entre os eixos x_{i-1} e x_i sobre o eixo z_{i-1} . É positivo no sentido horário.
- d_i : é a distância da origem do $(i-1)$ -ésimo sistema de coordenada para a interseção do eixo z_{i-1} com o eixo x_i ao longo do eixo z_{i-1} . Se o eixo z_{i-1} e o eixo x_i não se cruzarem, então é a distância perpendicular entre os eixos x_i e x_{i-1} .

- a_i : é a distância da origem do i -ésimo sistema de coordenada para a interseção dos eixos z_{i-1} e x_i ao longo do eixo x_i .
- α_i : é o ângulo de rotação medido do eixo z_{i-1} positivo, ou sua projeção paralela, para o eixo z_i positivo sobre o eixo x_i positivo, onde a direção positiva é o sentido horário.

Para uma junta de revolução, θ_i é a variável da junta e d_i é constante; para uma junta prismática, d_i é a variável e θ_i é constante.

Figura 6: Parâmetros para os elos e juntas.

O problema aqui proposto consiste portanto em: planejar a trajetória para o manipulador sair de uma posição de origem e chegar a uma posição meta, calcular a cinemática inversa e a dinâmica inversa para executar tal trajetória.

A rede utiliza os valores de posições espaciais x , y e z , os ângulos e os torques das seis juntas no instante atual para gerar as posições espaciais, os ângulos e os torques das seis juntas no instante seguinte. Para a rede gerar uma sequência, basta serem apresentados os estados inicial e final.

Para este problema também foram aplicados os quatro tipos de trajetórias ilustradas na Figura 2. Os aspectos considerados para avaliar o desempenho do modelo escolhido são os mesmos do caso anterior mais o funcionamento da rede em condições de ruído e/ou para trajetórias não-treinadas.

Os dados para o treinamento da rede foram gerados por um simulador. Trata-se da ferramenta *ROBOTICS* para o MATLAB [6] que fornece funções para transformação homogênea, cinemática direta e inversa, geração de trajetórias, dinâmica direta e inversa e animação gráfica. Esta ferramenta é útil tanto para simulações como para analisar resultados de experimentos com robôs reais. A ferramenta *ROBOTICS* utiliza um método geral para calcular a cinemática e a dinâmica de manipuladores com elos seriais por descrição de matrizes. Estas matrizes podem ser criadas pelo usuário para qualquer manipulador com elos seriais. A ferramenta já traz estas matrizes para os manipuladores PUMA 560 e Stanford.

2.2.1. PUMA 560

O manipulador robô PUMA (Programmable Universal Machine Assembly) [5] [2] é provavelmente o robô mais comum em laboratórios de universidades e um dos mais comuns em linhas de montagem. O modelo utilizado nesta dissertação é o PUMA 560, que é um robô manipulador com seis juntas rotacionais arranjadas sequencialmente. As seis juntas são geralmente denominadas como: corpo (junta 1); ombro (junta 2); cotovelo (junta 3); rotação de pulso (junta 4); dobra de pulso (junta 5) e *flange* (junta 6). Cada membro do braço mecânico é conectado ao outro por uma destas juntas, similares ao tronco e braço humanos.

Este manipulador possui seis graus de liberdade, três para as posições espaciais x , y e z e três para orientação localizadas no efetuador. Isto torna o manipulador capaz de alcançar um ponto qualquer em seu espaço de trabalho por qualquer trajetória. O efetuador, localizado no final do manipulador, interage diretamente com o ambiente externo e pode ser uma chave de fenda, uma garra ou qualquer outra ferramenta. Os seis graus de liberdade são controlados por seis servomotores DC. A Figura 7 apresenta os eixos de coordenadas para cada junta do PUMA 560.

É importante destacar que o manipulador PUMA 560 que será simulado pela ferramenta *ROBOTICS* do *software* MATLAB tem seus parâmetros de cinemática e dinâmica apresentados nas Tabelas 1, 2, 3 e 4 que estão no Apêndice I.

Figura 7: Detalhes dos eixos de coordenadas para o PUMA 560 mostrados aqui com seus ângulos na posição zero.

3. REVISÃO BIBLIOGRÁFICA

Este capítulo define brevemente redes neurais e introduz os mais importantes modelos de redes neurais para lidar com seqüências temporais. Ao final do capítulo, o leitor terá conhecido as principais características do neurônio artificial e da classe de redes neurais que serão empregadas nos modelos a serem propostos.

3.1 Redes Neurais

Redes neurais [47] são sistemas que anexam conhecimento através da extração de regularidades de um conjunto de dados. Redes neurais viabilizam a transferência de conhecimento para o computador por meio da apresentação de exemplos.

Os modelos de redes neurais visam replicar performance semelhante a de organismos inteligentes na resolução de problemas através da construção de sistemas computacionais inspirados no funcionamento do cérebro destes organismos. Desta maneira, uma rede neural pode ser vista como um conjunto de elementos processadores simples, baseados em neurônios biológicos, que são ligados uns aos outros através de conexões, análogas às sinapses. Estas conexões guardam o "conhecimento" da rede e os diversos padrões de atividade expressam os vários objetos representados pela rede. Em outras palavras, as conexões fazem o papel de memória de longo prazo enquanto que o estado de ativação das unidades de uma rede realizam o papel de memória de curto prazo. O conhecimento da rede é adquirido por meio de um processo de treinamento no qual, em suas versões mais básicas, apenas as conexões entre as unidades são variadas através das mudanças de pesos.

Devido à complexidade e à diversidade das propriedades dos neurônios biológicos, a tarefa de expressar suas características complexas em um modelo é

muito difícil e ainda restam muitas características, em especial a dinâmica destes neurônios, por serem melhor entendidas. O nível de detalhes que se deseja estudar, associado às dificuldades acima influenciam no aparecimento de um grande número de abstrações do neurônio real.

Figura 8: Modelo simplificado de neurônio utilizado em redes neurais artificiais.

Um modelo de neurônio muito utilizado dentro do paradigma rede neural é apresentado na Figura 8 [14] [38]. O neurônio recebe informações vindas de outros neurônios ou do meio externo. Estas informações passam pela sinapse que as ponderam e chegam ao corpo celular através dos dendritos. No corpo celular ocorre todo o processamento das informações de entrada do neurônio. A soma ponderada destas entradas constitui o argumento de uma função de ativação não-linear. O neurônio dispara se a soma ponderada destas entradas exceder um certo limiar, ω_o . A saída deste neurônio chega a outros neurônios ou ao meio externo através do axônio. Esta é uma aproximação simples, mas muito útil, do neurônio biológico. Matematicamente, a função de ativação do neurônio pode ser definida como:

$$y(t) = f\left(\sum_{i=1}^n \omega_i x_i(t) - \omega_o\right) \quad (3.1.1)$$

onde $x_1(t), \dots, x_n(t)$ representa a entrada do neurônio em questão, $\omega_1, \dots, \omega_n$ são os pesos sinápticos, $y(t)$ é a saída neural e $f(.)$ é qualquer função de ativação linear ou

não-linear com limiar ω_o . e $\sum_{i=1}^n \omega_i x_i(t)$ é a soma ponderada das entradas do

neurônio i.

Algoritmos de redes neurais podem ser supervisionados ou não-supervisionados. No primeiro caso, utiliza-se um “professor”. Durante o treinamento são fornecidos os pares entrada-saída e os pesos, os valores numéricos das conexões, são ajustados como função do erro entre a saída desejada e a saída obtida pela rede em treinamento. No caso dos algoritmos não-supervisionados, a própria rede se encarrega de mapear um espaço de entrada em um espaço de saída, auto-arranjando seus pesos para isto. Os algoritmos supervisionados são normalmente mais usados em processos de engenharia, pois eles podem garantir uma resposta desejada.

Os neurônios utilizados nos modelos propostos nesta dissertação serão iguais ao ilustrado na Figura 8, sendo $f(\cdot)$ a função sigmoidal para o problema de produção de séries temporais discretas ou a função tangente hiperbólica para o problema de produção de séries temporais contínuas. O algoritmo utilizado para o treinamento destes modelos será o de retropopagação, o algoritmo supervisionado mais utilizado.

3.2 Redes Neurais Recorrentes

Redes neurais podem ser classificadas de duas formas quanto ao fluxo de informação: com alimentação direta e recorrente. Redes neurais com alimentação direta são aquelas em que toda informação flui em uma mesma direção na rede: da entrada para a saída. Tais redes não têm memória dinâmica, já que sua resposta depende somente da entrada atual e dos valores dos pesos. Por não possuírem nenhuma realimentação, redes com alimentação direta são estáveis caso o problema tenha solução. Redes neurais de alimentação direta podem aproximar funções não-lineares para um grau de precisão desejado, o que tem feito muitos pesquisadores utilizá-las para modelar sistemas dinâmicos. Entretanto, estas redes apresentam certas limitações quando utilizadas com esta finalidade, como por exemplo, não consideram seu histórico, ou seja, não levam o passado em conta para decisões no presente.

Redes neurais recorrentes possuem conexões que permitem fluxo de informação em ambos os sentidos entre um par de unidades, e mesmo de uma unidade para ela mesma. Estas realimentações implicam que a rede absorve características de memória local. Tais redes convergem mais lentamente pois passam

por um processo iterativo para calcular sua saída. Para uma rede estável, sucessivas iterações produzem cada vez menos variações na saída até eventualmente as saídas se tornarem constantes. Em algumas situações, o processo nunca converge, tais redes são ditas instáveis. Redes neurais recorrentes são particularmente apropriadas para modelamento de sistemas de controle e aplicações de filtragem. Estas redes são importantes porque muitos dos sistemas do mundo real são sistemas dinâmicos não-lineares. Exemplos destes sistemas de controle incluem dinâmica direta ou inversa de sistemas tal como aviões, foguetes, naves espaciais e robôs.

Redes neurais recorrentes podem ser divididas em completamente recorrentes ou parcialmente recorrentes [17]. Redes neurais completamente recorrentes são aquelas em que todas as unidades da rede estão conectadas a todas as suas outras unidades. Redes neurais parcialmente recorrentes estão entre as arquiteturas tipo perceptron multicamadas somente com alimentação direta e uma arquitetura de rede completamente recorrente. Isto é, suas conexões são predominantemente de alimentação direta, mas apresentam conexões de realimentação cuidadosamente escolhidas.

Dentre as arquiteturas parcialmente recorrentes, um grupo tem chamado bastante atenção: as arquiteturas localmente recorrentes globalmente com alimentação direta (LRGF). Nestes modelos, a realimentação ocorre dentro da própria unidade de processamento, ou seja, cada unidade de processamento realimenta a si própria e não às outras unidades da rede [52].

Há três métodos nos quais realimentação local pode ser incorporada.

- Realimentação local da soma ponderada - A realimentação pode ser versões atrasadas da soma ponderada das entradas, ou combinação de versões atrasadas desta e versões atrasadas de variáveis de entrada.
- Realimentação local de sinapse - Cada sinapse pode incorporar uma estrutura de realimentação. Esta estrutura de realimentação pode ser na forma de um filtro de resposta de impulso finito, ou um filtro de resposta de impulso infinito.

- Realimentação local de saída - A entrada de cada unidade recebe seus valores atrasados da saída.

Há duas classes principais de algoritmo de treinamento: algoritmos de aprendizagem com ponto fixo e com ponto não-fixo [41]. Os algoritmos de aprendizagem com ponto fixo envolvem redes que convergem para pontos fixos estáveis. Um conjunto de estados iniciais ou uma entrada constante externa apresentados à rede produz um estado de saída. Esta abordagem é freqüentemente aplicada em tarefas de satisfação de restrições e memória associativa. O algoritmo de retropropagação recorrente [1] [42], a máquina de Boltzmann determinística [19] e modelos que transformam seqüência temporal em espacial [49] são exemplos típicos desta classe de algoritmos. Os algoritmos de aprendizagem de pontos não-fixos empregam minimização de erro como função do comportamento temporal da rede. Alguns exemplos desta classe de modelos são o algoritmo de retropropagação através do tempo (BPTT) [45] [46], a arquitetura Williams-Zipser [55] [58] e o modelo Elman-Jordan [9] [20]. Os dois últimos modelos são uma versão discreta do BPTT em que a história temporal é cortada em um ou dois passos de tempo. Nesta dissertação os modelos propostos vão lidar apenas com algoritmos de aprendizagem com pontos não-fixos.

3.2.1 Transformação de seqüência temporal em seqüência espacial.

Redes neurais parcialmente recorrentes têm sido muito utilizadas para reconhecer e reproduzir seqüências temporais ou espaciais. Este tipo de recorrência permite à rede lembrar o passado recente, e não adiciona muita complexidade ao treinamento.

Em modelos de computação paralela, o processamento de entradas seqüenciais tem sido executado de várias formas. A solução mais comum é transformar a representação temporal em representação espacial para a utilização do método de retropropagação convencional para treinar a rede. Um exemplo disto é o NET-talk [49] que visa treinar uma rede com alimentação direta para pronunciar

textos em inglês. Neste caso, a entrada consiste em sete caracteres consecutivos de um mesmo texto apresentados em uma janela móvel que gradualmente varre o texto. Esta representação espacial do tempo o trata como uma parte explícita da entrada. Elman [9] aponta alguns problemas para esta abordagem. Um destes problemas é que ela requer que haja alguma interface com o mundo que armazene todas as entradas necessárias, a fim de que possam ser apresentadas todas de uma vez. Outro problema é que o registrador móvel impõe um limite rígido na duração dos padrões, além disto, sugerem que todos os vetores de entrada tenham o mesmo tamanho. O problema considerado mais sério é que não é fácil distinguir posição temporal relativa de posição temporal absoluta. Por exemplo, considere os dois vetores:

$$[0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0] \text{ e } [0\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0]$$

Estes dois vetores parecem ser exemplos do mesmo padrão, porém deslocados no espaço. Entretanto, como a interpretação geométrica destes vetores evidencia claro, os dois padrões são muito diferentes e estão espacialmente distantes. Modelos conexionistas podem ser treinados para tratar estes dois padrões como similares, mas a similaridade é consequência do aprendizado externo e não da similaridade das estruturas dos padrões. Além disto, a similaridade desejada não generaliza para novos padrões. Esta falha é séria quando se está interessado em padrões em que a estrutura temporal relativa seja preservada, decorrente do deslocamento temporal absoluto.

Uma melhor abordagem seria representar o tempo implicitamente ao invés de explicitamente. Neste caso, utiliza-se o tempo pelo efeito que tem no processo e não como uma dimensão adicional da entrada. Isto permite ao processo considerar propriedades que são responsáveis pela dinâmica da sequência temporal, ou seja, dá-se memória à rede.

3.2.2 O algoritmo de retropropagação através do tempo e algumas de suas derivações.

O algoritmo de retropropagação através do tempo (BPTT) [45] [46] é um algoritmo de aprendizagem para o treinamento de redes neurais completamente recorrentes. A ativação da j -ésima unidade de processamento $o_j(t+1)$ é função das saídas anteriores de cada unidade da rede $o_i(t)$ e da última entrada $i_i(t)$, se houver. Assim, a regra de atualização é dada por:

$$o_i(t+1) = f\left(\sum_{j=1}^{nu} \omega_{ij} o_j(t) + i_i(t)\right) \quad (3.2.2.1)$$

BPTT foi originalmente proposto como tendo ativação contínua e atualização discreta. Na fase de treinamento a rede é replicada T vezes, onde T é o número de passos de tempo da seqüência. Este processo desdobra a rede recorrente em uma rede de alimentação direta para ser treinada. A Figura 9 mostra a idéia para uma rede recorrente com duas unidades e $T = 4$ passos de tempo. Note que as conexões ω_{ij} de O_j^t para O_i^{t+1} são independentes do tempo t ; os mesmos valores de pesos devem ser usados em cada camada. A rede resultante da expansão é estritamente de alimentação direta podendo ser treinada por uma forma levemente modificada do algoritmo de retropropagação.

Figura 9: Diagramas do modelo BPTT: (a) Topologia do modelo; (b) Modelo desdobrado com alimentação direta para fins de treinamento.

O principal problema com o BPTT é que necessita de grande recurso computacional. Necessidade de armazenamento, tempo de cálculo por simulação e número de exemplos de treinamento necessários são todos aumentados pela multiplicação das unidades. Para grandes seqüências ou para seqüências de tamanho desconhecido o uso deste algoritmo torna-se impraticável. Note que a multiplicação aplica-se apenas para a fase de treinamento.

Quatro versões do algoritmo BPTT foram descritas por Williams e Zipser [57]: BPTT por época, onde o treinamento é segmentado em épocas e todo o histórico de entradas, estados e alvos sobre um intervalo de época são salvos; BPTT em tempo real, onde os pesos mudam em cada passo de tempo enquanto a rede funciona continuamente e todo o histórico das entradas e estados são salvos; BPTT truncado, que é uma aproximação para BPTT em tempo real com histórico limitado em que informações relevantes são salvas para um número fixo k de passos de tempo e qualquer informação fora deste intervalo não é considerada explicitamente; e BPTT por época e truncado, que é obtido combinando aspectos do BPTT por época e do BPTT truncado. Em *Batch* BPTT (BBPTT), o gradiente para cada peso é calculado para cada padrão como no BPTT truncado e então toma um valor médio para o conjunto de treinamento completo. *Quickprop* através do tempo (QPTT) é outro algoritmo de aprendizagem para redes neurais completamente recorrentes que calcula o gradiente como em BBPTT, mas os pesos são adaptados usando uma regra de atualização mais eficiente: *quickprop* [11] que usa informação sobre a curvatura da superfície do erro para aumentar a velocidade de aprendizagem. A regra de atualização *quickprop* trata-se de uma variação heurística do algoritmo de retropropagação que emprega momentum dinâmico.

Williams e Zipser [55] [56] introduziram um método que permite que redes recorrentes aprendam tarefas que requerem retenção de informação sobre um período de tempo sem multiplicar as unidades. Uma versão desta regra pode funcionar *on-line*, aprendendo enquanto seqüências estão sendo apresentadas. Isto evita que seja armazenada grande quantidade de informação durante o treinamento. Esta regra é conhecida como aprendizagem recorrente em tempo real (RTRL). Uma modificação freqüentemente encontrada é o uso do professor forçante (*teacher forcing*).

Professor forçante é um processo que substitui o valor obtido na saída da rede sempre que este estiver errado pelo valor desejado sendo este conhecido. Isto deve ser feito depois de calcular o erro e o conjunto de derivadas. O procedimento professor forçante mantém a rede próxima do caminho desejado e geralmente parece aumentar consideravelmente a velocidade de treinamento.

Vários exemplos investigados por Williams e Zipser [56] demonstraram o poder e generalidade do método RTRL. Este trabalha bem para tarefas de reconhecimento de seqüências, embora isto possa ser realizado também por redes mais simples.

Na tentativa de criar uma rede neural capaz de tomar decisões considerando um contexto, Elman [9] propôs uma rede neural parcialmente recorrente, que é capaz de identificar e classificar estruturas temporais. Elman propôs uma arquitetura com três camadas (Figura 10) que consiste de uma camada de entrada com n_i unidades de entrada e n_c unidades de contexto, uma camada escondida com n_e unidades e uma camada de saída com n_s unidades, neste modelo tem-se $n_c = n_e$. Todas as unidades da camada de entrada são conectadas a todas as unidades da camada escondida. O mesmo ocorre entre as unidades da camada escondida e as da camada de saída. A informação é propagada da camada de entrada para a camada de saída. O estado de ativação de cada unidade da camada escondida é copiado na posição correspondente nas unidades de contexto. Neste modelo, as conexões de alimentação direta são treináveis enquanto as conexões de realimentação são fixas e iguais a 1. As unidades de contexto fornecem memória à rede.

Figura 10: Modelo de Elman. Neste modelo e em todos os outros de agora em diante, as linhas em negrito denotam conexões todos-para-todos e as linhas normais conexões um-para-um.

A ativação em qualquer unidade de escondida ou de saída é dada por:

$$y_i(t) = f\left(\sum_j \omega_{ij} \lambda_j(t) + \omega_{0i}\right) \quad (3.2.2.2)$$

onde $f(.)$ é a função sigmoidal, $\lambda_j(t)$ é o estado de ativação da unidade j de entrada ou da unidade j escondida.

O estado de ativação da i -ésima unidade de contexto é dado por:

$$c_i(t+1) = h_i(t) \quad (3.2.2.3)$$

onde $h_i(.)$ representa a ativação das unidades da camada escondida.

Os valores de ativação inicial das unidades da camada de contexto são geralmente colocados em 0.5.

Kremer [28] discute algumas características da rede de Elman e prova que seu poder computacional é tão grande quanto a de máquinas de estados finitos.

Redes neurais são tipicamente de tamanho finito e são implementadas usando aritmética de precisão fixa, logo sua memória é finita. Estas duas características são compartilhadas por máquinas de estados finitos que têm um grande poder computacional. Assim tenta-se demonstrar que uma rede de Elman pode possuir comportamento análogo a uma máquina de estados finitos.

Primeiramente Kremer prova que um único nó na rede de Elman pode executar a lógica OU para as ativações de seus nós de entrada. Em seguida, descreve como um nó pode calcular uma combinação especial de lógicas E's e Ou's. Depois prova-se que uma rede de Elman pode calcular a seqüência de transição de estado de qualquer máquina de estados finitos. Finalmente prova-se que uma rede de Elman pode calcular qualquer saída de uma máquina de estados finitos. Kremer provou que embora a saída da rede de Elman não seja realimentada para sua entrada, o estado escondido é suficiente para calcular o próximo estado.

Jordan [20] introduziu uma rede (Figura 11) contendo conexões de realimentação que foram usadas para associar um padrão de entrada a um padrão de saída ordenado serialmente. As conexões de realimentação permitem que as unidades

de contexto da rede recebam as saídas anteriores dela e assim o comportamento da rede pode ser influenciado por elas. Estas recorrências provêm memória à rede.

Figura 11: Modelo proposto por Jordan.

Jordan propôs uma arquitetura com a mesma topologia de realimentação direta da rede de Elman, entretanto as unidades de contexto recebem informações das unidades de saída e delas mesmas. Os pesos de realimentação não são treináveis. O estado de ativação da i -ésima unidade de contexto é definida como:

$$c_i(t+1) = \beta c_i(t) + o_i(t) = \sum_{k=0}^t \beta^{(t-k)} o_i(t) \quad (3.2.2.4)$$

onde $o_i(t)$ são as unidades de saída e β é a força das conexões de realimentação. É requerido $\beta < 1$. Logo, fazendo β próximo de 1, a memória do passado pode se prolongar mais, mas isto resulta na perda de sensibilidade a detalhes das novas entradas.

Em resumo, a arquitetura Elman-Jordan é caracterizada por um conjunto extra de unidades de processamento nas quais a rede copia os estados de ativação atrasados da camada escondida ou de saída. Estas conexões de realimentação são fixas com valor igual a 1, enquanto as conexões de alimentação direta são treinadas pelo algoritmo de retropropagação e atualizadas sincronamente. Para o modelo de Jordan, há também conexões de cada unidade de contexto para ela mesma. Esta topologia pode falhar para alcançar estabilidade, apresentar um processo de convergência muito lento ou decrescer seu erro de maneira dente de serra. A fim de superar algumas destas limitações foram propostas arquiteturas localmente recorrentes globalmente de alimentação direta.

3.2.3 Arquiteturas LRGF

Tsoi e Back [52] apresentam várias arquiteturas de redes neurais localmente recorrentes globalmente com alimentação direta, propostas por diferentes grupos de pesquisadores, salientando suas diferenças e propondo uma estrutura geral de rede LRGF que inclui a maioria das arquiteturas abordadas. Algumas arquiteturas típicas foram propostas por Back e Tsoi [4] e Frasconi, Gori e Soda [12].

Back e Tsoi [4] introduziram uma arquitetura LRGF com realimentação local de sinapse. Nesta arquitetura, em vez de uma sinapse com pesos constantes, os autores introduziram uma sinapse com uma função de transferência linear com pólos e zeros. As realimentações locais de sinapse localizam-se na camada escondida. A função de ativação da i -ésima unidade de processamento na camada escondida é:

$$H_i(t) = f \left(\sum_{j=0}^n G_j(z^{-1}) x_j(t) \right) \quad (3.2.3.1)$$

com $x_j(t)$, $j = 1, 2, \dots, n$ é o conjunto de entradas das unidades de processamento da camada anterior, x_0 é o limiar, $G_j(z^{-1})$ é uma função de transferência linear dada por:

$$G_j(z^{-1}) = \frac{\sum_{i=0}^{n_z} b_i z^{-i}}{\sum_{i=0}^{n_p} a_i z^{-i}} \quad (3.2.3.2)$$

onde $z^{-i} x(t) = x(t-i)$; a_i, b_i são constantes diferentes de zero, e n_p, n_z são o número de pólos e zeros respectivamente.

Frasconi et al. [12] introduziram uma arquitetura LRGF com realimentação local de saída, ou seja, a realimentação local é tomada depois da função não-linear.

$$y_i(t) = f \left(\sum_{i=1}^m k_i y_i(t-1) + \sum_{j=0}^n \omega_{ij} x_j(t) \right) \quad (3.2.3.3)$$

onde x_j são entradas das camadas anteriores à camada em que se está calculando a ativação ou entradas para a rede, k_i e ω_{ij} são pesos e m e n são o número de unidades nas camadas consideradas.

A Figura 12 mostra um modelo geral de rede LRGF que inclui a arquitetura com realimentação local de sinapse, de ativação $q(t)$ e também de saída $y(t)$. O modelo geral de rede LRGF pode ser considerado como uma generalização do

clássico neurônio de McCulloch-Pitt [36] de um ponto de vista entrada-saída. As entradas são $x_1, \dots, x_n$ e a saída é $y(t)$. Em vez de pesos sinápticos constantes, tem-se sinapses que são modificáveis por funções de transferência lineares. Além disto, a realimentação da saída e de ativação são também modeladas por funções de transferência linear. Este neurônio generalizado pode então ser usado em uma arquitetura do tipo perceptron multicamadas com alimentação direta.

Figura 12: Uma topologia geral para redes LRGF.

Stornetta et al. [51] e Mozer [39] propuseram modelos com realimentação local de saída. Stornetta et al. apresentaram um modelo de alimentação direta com três camadas em que as unidades de contexto, formando a camada de entrada, são realimentadas por suas próprias saídas (Figura 13).

Figura 13: Modelo proposto por Stornetta et al..

Neste caso, a recorrência pré-processa o sinal de entrada a fim de considerar o passado do sinal de entrada no contexto presente, permitindo à rede reconhecer e distinguir diferentes seqüências. Assim, a ativação da i -ésima unidade de contexto é

influenciada por sua própria saída e por estímulos $s_i(t)$ fornecidos pelo ambiente. Isto é apresentado na seguinte equação:

$$c_i(t+1) = \beta c_i(t) + s_i(t) \quad (3.2.3.4)$$

Este modelo mostra-se menos eficiente do que os modelos propostos por Elman e Jordan para gerar ou reproduzir seqüências [17].

Figura 14: Modelo proposto por Mozer.

Mozer introduziu uma arquitetura com realimentação local de saída que é formada por quatro camadas de alimentação direta (Figura 14). Esta parece a princípio similar ao modelo proposto por Stornetta et al., mas cada unidade de contexto tem conexões ajustáveis um-para-um, o que permite a rede considerar a história do contexto. As unidades da camada de entrada e da camada de contexto são completamente conectadas. As unidades de contexto treinadas atuam como um integrador para considerar os estados de contexto anterior. As conexões de realimentação treináveis permitem ao modelo encontrar uma taxa de decaimento adequada para a camada de contexto.

Mozer propôs uma regra de aprendizagem similar a de Williams e Zipser. A ativação da i -ésima unidade de contexto é dada por:

$$c_i(t+1) = \beta c_i(t) + f\left(\sum_{j=1}^{n_i} \omega_{ij} i_j(t)\right) \quad (3.2.3.5)$$

onde n_i é o número de unidades da camada de entrada.

As arquiteturas descritas devem apresentar desempenhos diferenciados para tarefas distintas como descrito a seguir.

3.2.4 Comparação Entre Desempenhos da Arquiteturas

Tsoi e Back [52] utilizaram as arquiteturas de Back-Tsoi, Frasconi-Gori-Soda, Williams-Zipser e a arquitetura perceptron multicamadas usando uma janela de entrada atrasada para resolver um problema de predição de um passo a frente de uma série temporal. No caso, a série temporal é uma expressão de fala. Por causa das diferenças entre as arquiteturas, é muito difícil encontrar uma base para comparação. Mesmo assim, foi escolhida a predição mencionada onde a arquitetura que tiver a menor variância do erro na predição é considerada como a melhor arquitetura.

Os resultados obtidos foram:

- A arquitetura Frasconi-Gori-Soda tem melhor desempenho, seguida da MLP.
- A arquitetura Williams-Zipser tem o pior desempenho
- A arquitetura Back-Tsoi e a MLP parecem ter variâncias do erro de predição similares.
- A arquitetura Back-Tsoi e até certo ponto a arquitetura Frasconi-Gori-Soda parecem ter uma componente de alta frequência no erro de predição próximo ao pico das séries temporais.
- A arquitetura de Frasconi-Gori-Soda e a MLP parecem ter desempenhos muito similares.
- A arquitetura Williams-Zipser parece levar mais iterações para convergir do que as outras arquiteturas.
- O comportamento da arquitetura Back-Tsoi parece ser diferente da arquitetura de Frasconi-Gori-Soda.

Estes experimentos sugerem que a arquitetura completamente recorrente de Williams-Zipser apresenta treinamento mais longo em relação às arquiteturas LRGF, e pior desempenho do que as outras três arquiteturas quando aplicadas ao problema de reconhecimento de seqüências. Os testes sugerem também que as informações realimentadas a partir da saída das unidades são qualitativamente melhores do que as

informações realimentadas a partir das sinapses das unidades de processamento, visto que a arquitetura de Frasconi-Gori-Soda apresentou melhor desempenho do que a arquitetura de Back-Tsoi.

Ludik et al. [32] apresentam uma investigação experimental de natureza qualitativa comparando redes neurais completamente recorrentes usando algoritmos de aprendizagem como o algoritmo de retropropagação através do tempo (BPTT), *Batch* BPTT (BBPTT), *Quickprop* através do tempo (QPTT) e aprendizagem recorrente em tempo real (RTRL) e também as redes parcialmente recorrentes de Jordan e Elman em quatro problemas de teste de desempenho:

- Teste de desempenho de gramática para a detecção de três zeros em uma sequência de bits.
- Identificação de planta não-linear.
- Emulação da máquina de Turing.
- Modelamento de uma coluna de destilação de um processo químico do mundo real.

Nos diferentes testes de desempenho realizados para as várias arquiteturas completa ou parcialmente recorrentes, ficou evidente que QPTT se apresenta melhor entre as redes completamente recorrentes, enquanto a rede parcialmente recorrente de Elman não somente superou a rede de Jordan em todos os testes, mas também mostra-se melhor que o BPTT truncado, BBPTT e RTRL.

Os resultados obtidos comparando os diferentes tipos de redes recorrentes sugerem que tanto as redes LRGF como as parcialmente recorrentes apresentam bom desempenho quando aplicadas a problemas de reconhecimento de sequências temporais. Já as arquiteturas completamente recorrentes mostram-se mais lentas para treinamento e, em geral, têm desempenho inferior às parcialmente recorrentes. A arquitetura proposta por Jordan alcança melhor desempenho na produção de sequências temporais. Isto sugere que realimentação intercamada é importante para produção de sequências temporais.

Estes resultados são motivadores para propor redes parcialmente recorrentes por sugerirem melhor desempenho e maior velocidade de treinamento com relação as arquiteturas completamente recorrentes. Todos os modelos de redes parcialmente recorrentes propostos nesta dissertação apresentam conexões de realimentação intercadas, pois elas parecem ser de fundamental importância para a produção de seqüências temporais. Além disto, as arquiteturas LRGF parecem ter desempenho bom na tarefa de reconhecimento de seqüências logo este tipo de realimentação deve ser considerado para resolver conflitos. A determinação de um algoritmo de treinamento segue a escolha da topologia da rede.

3.2.5 Algoritmo de treinamento

Nerrand et al. [40] mostram a importância da escolha de um algoritmo apropriado para o treinamento de uma rede neural recorrente em um dado problema, com base no modo como ruídos aleatórios interferem no processo.

Três famílias de algoritmos de treinamento são consideradas: indiretos, semi-diretos e diretos. Estas diferem entre si pela escolha dos valores dos estados de entrada. A dinâmica de uma rede recorrente discreta no tempo pode ser descrita por N equações de diferenças de primeira ordem envolvendo N variáveis, chamadas variáveis de estado, em adição a M variáveis de entrada. Os algoritmos indiretos são aqueles em que as variáveis de estado da rede não são forçadas a serem iguais aos valores desejados das variáveis de estado do processo a ser modelado pela rede. Isto é, as variáveis de estado encontradas pela rede são mantidas. Para os algoritmos semi-diretos apenas a variável de estado referente ao tempo atual é forçada a ser idêntica ao estado desejado. Algoritmos diretos, também conhecidos como professor forçante (*teacher forcing*), são aqueles em que todas as variáveis de estado são forçadas a serem idênticas às desejadas.

Nerrand et al. apresentam três modelos de processos com realimentação, diferenciados pela inserção do ruído em locais diferentes do processo e mostram para cada caso qual tipo de algoritmo de treinamento é o mais apropriado. Os modelos utilizados foram: modelo erro de saída (Figura 15.a), modelo NARMAX (*Non-linear*

Auto-Regressive Moving Average with Exogeneous Inputs - Figura 16.a) e o modelo NARX (*Non-linear Auto-Regressive with Exogeneous Inputs* - Figura 17.a). Para cada processo, um modelo predictor é proposto pelos autores. O comportamento dinâmico deste modelo deve ser o mais próximo possível do comportamento do processo.

Figura 15: (a) Estruturas de modelos de processo sobre a hipótese erro de saída e (b) predictor neural associado.

Para o modelo erro de saída, é assumido que o estado do processo é função do valor de entrada atual e de uma determinada quantidade de entradas anteriores e também de uma determinada quantidade das últimas variáveis de estado. A saída do processo é definida como o estado atual adicionado a um ruído branco. As variáveis de estado $u_p(t)$ e a saída $y_p(t)$ seguem as equações:

$$u_p(t) = \Phi[U_p(t-1), X(t-1)], \quad (3.2.5.1)$$

$$y_p(t) = u_p(t) + w(t), \quad (3.2.5.2)$$

onde: $U_p(t-1) = \{u_p(t-1), u_p(t-2), \dots, u_p(t-N)\}$ é um vetor com as últimas N variáveis de estado, $X(t-1) = \{x(t-1), x(t-2), \dots, x(t-M)\}$ é um vetor com os M últimos valores de entrada e $w(t)$ é uma sequência de ruído branco.

Este processo origina um previsor neural cuja saída $y_p(t)$ é tal que $y_p(t) - y(t) = w(t)$, ou seja, a saída do processo sem o ruído é função de saídas anteriores e de entradas anteriores como:

$$y(t) = \Phi[y(t-1), \dots, y(t-N), X(t-1)] \quad (3.2.5.3)$$

Figura 16: (a) Estruturas de modelos de processo sobre a hipótese NARMAX e (b) previsor neural associado.

Para o modelo NARMAX, o estado do processo é função de variáveis de estados anteriores, entradas anteriores e ruídos anteriores mais o ruído no tempo atual. Este modelo obedece as seguintes equações:

$$u_p(t) = \Phi[U_p(t-1), X(t-1), W(t-1)] + w(t), \quad (3.2.5.4)$$

$$y_p(t) = u_p(t). \quad (3.2.5.5)$$

onde: $U_p(t-1) = \{u_p(t-1), u_p(t-2), \dots, u_p(t-N)\}$ é o vetor com as N últimas variáveis de estado, $W(t-1) = \{w(t-1), w(t-2), \dots, w(t-P)\}$ é um vetor com os P últimos valores para o ruído e $X(t-1) = \{x(t-1), x(t-2), \dots, x(t-M)\}$ é um vetor com os M últimos valores de entrada.

A saída $y(k)$ do previsor associado é definida por:

$$y(t) = \Phi[Y_p(t-1), X(t-1), e(t-1), \dots, e(t-P)] \quad (3.2.5.6)$$

onde: $e(t) = y_p(t) - y(t)$ e $Y_p(t-1) = \{y_p(t-1), y_p(t-2), \dots, y_p(t-N)\}$.

Figura 17:(a) Estruturas de modelos de processo sobre a hipótese NARX, (b) previsor neural associado.

Para o modelo NARX, é assumido que o processo obedece as equações que seguem:

$$u_p(t) = \Phi[U_p(t-1), X(t-1) + w(t)] \quad (3.2.5.7)$$

$$y_p(t) = u_p(t). \quad (3.2.5.8)$$

A saída $y(t)$ do previsor é obtida por:

$$y(t) = \Phi[Y_p(t-1), X(t-1)] \quad (3.2.5.9)$$

com $Y_p(t-1) = \{y_p(t-1), y_p(t-2), \dots, y_p(t-N)\}$.

Os autores mostram que algoritmos diretos são apropriados para o modelamento de sistemas dinâmicos de baixo ruído, ou para sistemas em que perturbações aleatórias podem ser consideradas como ruído branco adicionados às variáveis de estado do modelo, pois para tais modelos é necessário se ter os valores desejados para todas as variáveis de estado do sistema. Algoritmos indiretos são apropriados para prever a saída de sistemas em que perturbações aleatórias podem ser consideradas como ruído branco adicionados à saída do modelo. Algoritmos semi-diretos são úteis quando um modelo de erro de saída descreve o processo apropriadamente, porém quando o correspondente previsor é instável. Portanto,

algoritmos diretos são mais apropriados para o treinamento de redes neurais para prever os modelos NARX e NARMAX enquanto algoritmos indiretos são mais apropriados para treinar redes neurais para prever o modelo erro de saída

Para os problemas tratados nesta dissertação, as variáveis de estado são valores atrasados da saída e para todas elas existe um valor desejado associado portanto, opta-se por um algoritmo de aprendizagem direto, ou seja, professor forçante.

3.3 Redes Neurais para Lidar com Planejamento e Controle de Trajetória de Braço Mecânico.

O problema de planejamento e controle de trajetória do braço do robô tem diferentes abordagens em termos de redes neurais. Contudo, o aprendizado sempre ocorre através de experiência, isto é, por meio de exemplos. A escolha do comportamento é feita pelo robô com base em exemplos. Este mecanismo substitui um conjunto de comportamentos pré-selecionados. A escolha do comportamento é alcançada ao final do processo de treinamento da rede, que mapeia o espaço de entradas do sistema em seu espaço de saída.

Figura 18: Robô manipulador bidimensional com dois graus de liberdade.

Os primeiros modelos utilizando redes neurais em robótica mantiveram o problema e o modelo o mais simples possível. Tratou-se apenas o problema de cinemática para um robô bidimensional com dois graus de liberdade (Figura 18).

Uma rede neural de alimentação direta com método de aprendizagem supervisionado foi proposta por Josin [21]. Ele simulou um manipulador bidimensional com dois elos de comprimento L_1 e L_2 e ângulos de juntas θ_1 e θ_2 .

As equações de cinemática direta e inversa do manipulador geraram as relações entre a posição do efetuador (x, y) e os ângulos das juntas do robô (θ_1, θ_2) .

Os resultados das equações de cinemática inversa foram ensinados a uma rede com duas unidades de entrada, trinta e duas unidades escondidas e duas unidades de saída. A rede foi treinada para um grid bidimensional 5x5. O treinamento, usando o algoritmo de retropropagação, foi curto e as respostas obtidas foram as desejadas. Esta rede é bastante limitada, pois entre outros fatores, o espaço de trabalho é muito pequeno e os resultados de cinemática inversa não são obtidos por tentativa e erro.

Para resolver o mesmo problema, Psaltis et al. [43] propuseram uma rede neural com alimentação direta auto-supervisionada que aprende através de experimentação em duas dimensões. Neste caso, as saídas desejadas da rede são geradas por três estratégias diferentes e não são conhecidas a priori como no caso passado.

A aprendizagem indireta, o primeiro método, utiliza uma rede neural para realizar o mapeamento dos pontos cartesianos do efetuador em ângulos das juntas do robô. Dado um par de coordenadas (x, y) , a rede produz os ângulos (θ_1, θ_2) . O manipulador move-se para esta posição de ângulo de juntas e gera, por sensoriamento, um novo par $(x, y)'$. Este novo par é apresentado à rede que produz $(\theta_1, \theta_2)'$. A rede tem seus parâmetros adaptados para minimizar o erro $(\theta_1, \theta_2)' - (\theta_1, \theta_2)$. Tal minimização não garante a minimização do erro total.

Na aprendizagem geral, o segundo método, Psaltis et al. empregaram o mesmo método anterior; contudo, a entrada para o robô, (θ_1, θ_2) , é fornecida pelo usuário. A rede interpola pontos intermediários.

O terceiro método, chamado de aprendizagem especializada, utiliza a matriz Jacobiana da função da planta para calcular sua variação quando seus parâmetros mudam. O valor do erro calculado usando-se o Jacobiano é retropropagado.

Trabalhos mais recentes aumentam o grau de complexidade do problema considerando a atuação em três dimensões e um braço de robô com três graus de liberdade.

Uma rede com alimentação direta em conjunto com uma estratégia de ajuste de entrada foi o método proposto por van der Smagt et al. [53]. Tal método não

demanda conhecimento prévio da planta. Os autores introduziram um sistema para coordenação mão-olho estática que é auto-adaptável em função de mudanças no meio-ambiente. O sistema de controle deve posicionar o efetuator sobre um dado alvo para coletá-lo. O controlador só lida com informações de posição, mapeando informações visuais atuais de posição em informações da posição a ser alcançada.

O modelo acima usou dois tipos de premissas sobre o robô. Primeiro, os autores assumiram que o robô é controlado especificando-se a mudança de ângulos de suas juntas e que os valores instantâneos destes ângulos são monitorados. Em segundo lugar, o sistema visual provê apenas informações sobre o objeto observado.

A entrada da rede é composta pelos dados visuais de observação do objeto alvo com respeito à posição alvo e pelos dados posicionais do braço mecânico. O controlador adaptável, a rede neural, aprende a gerar os diferenciais de posição que movimentam o efetuator na direção do objeto alvo. A transformação estabelecida neste caso é do tipo múltipla entrada para saída única, e é obtida através de treinamento supervisionado no qual posições ambíguas são deliberadamente evitadas.

Todas as abordagens até então só trataram do problema da cinemática. Alguns outros modelos também lidam com o problema da dinâmica.

Um modelo não-supervisionado foi proposto nos trabalhos de Ritter et al. [44] e Martinez et al. [34]. Eles utilizaram uma rede do tipo Kohonen [24] para controle de um braço com três graus de liberdade, o qual deve coletar um objeto no espaço tridimensional. Informações sensoriais foram fornecidas por duas câmeras fixas, que geravam as coordenadas do objeto alvo e do efetuator.

O movimento neste sistema foi dividido em duas etapas: um movimento grosseiro ou de aproximação da posição alvo e um movimento fino ou de ajuste. Na primeira etapa, a posição espacial do objeto é fornecida à rede do tipo Kohonen que ativa uma unidade num *lattice* tridimensional. A unidade ativa corresponde à posição do objeto no espaço de trabalho do robô. Cada unidade ativa tem um vetor $\vec{\theta}$ de ângulo das juntas e uma matriz Jacobiana da função da planta (A) associados a ela. Estes valores permitem o segundo movimento que corrige a discretização anterior por expansão em Taylor de primeira ordem. A aprendizagem da rede consiste em treinar os pesos, entre cada entrada e cada unidade do *lattice* como na rede de

Kohonen, e adaptar os valores de $\bar{\theta}$ e A por minimização do erro de posição do efetuador.

Em movimentos lentos, os servo-motores do braço devem mudar o torque individualmente. Contudo, para movimentos rápidos isto não é possível pois forças inerciais, centrífugas e giroscópicas produzem acoplamento complexo e não-linear das juntas. Neste caso, existe a necessidade de se considerar as equações Newtonianas de movimento do braço, ou seja, a dinâmica do braço. Ritter et al. resolveram este problema estendendo o algoritmo de Kohonen empregado na cinemática para a dinâmica do braço. Neste caso, o vetor velocidade do efetuador substitui o vetor posição no modelamento para fins de treinamento e funcionamento da rede.

Baseado no nível computacional de Marr [33], Kawato [23] propôs um modelo computacional para movimento voluntário de um braço mecânico multi-articulado para realização de uma tarefa. Kawato compara a solução por estratégia clássica de robótica com dois modelos de redes neurais para formação de trajetória e controle do movimento. Os modelos em redes neurais usaram resultados de experimentos comportamentais e conhecimento de fisiologia. Kawato propôs a resolução de problemas de trajetória, cinemática e dinâmica inversa por controle de movimentos voluntários. Este problema não tem solução única, logo o autor combinou um índice de performance compatível com esquemas computacionais, com algoritmos e modelos de redes neurais para realização do controle motor.

Kawato [22] apresentou evidências de que a trajetória desejada em coordenadas visuais e em coordenadas do corpo, assim como torque ativo, são internamente representados no cérebro. Deste modo, ele substitui a metodologia de robótica clássica por duas outras estratégias: (i) solução dos problemas de determinação de trajetórias, transformação de coordenadas e geração de comando motor em uma única etapa; (ii) resolução simultânea dos problemas de transformação de coordenadas e geração de comando motor, ficando a determinação de trajetória para uma primeira etapa.

A primeira estratégia emprega uma rede neural com módulos em cascata para determinação da trajetória do robô. O braço é treinado pelo algoritmo de

retropropagação através do tempo que produz caminhos ótimos para o braço, de acordo com o critério de mudança mínima de torque. Kawato construiu um modelo de rede neural invariante no tempo composto por “n” módulos de quatro camadas cada, dispostos em cascata. A entrada de cada módulo recebe o torque ótimo, a posição e a velocidade espacial do objeto num dado instante de tempo. A saída gera o valor das duas últimas variáveis de entrada no próximo instante de tempo. Os valores de saída de um bloco são alimentados no bloco posterior juntamente com o torque ótimo do instante de tempo seguinte. As correções dos pesos entre as camadas dos módulos são feitas pelo algoritmo de retropropagação comparando-se os valores de saída de cada bloco com os valores sensorizados no sistema físico. Wada e Kawato [54] discutiram as limitações deste modelo e propuseram uma nova rede baseada no mesmo critério, que não requer representação espacial do tempo ou algoritmo de retropropagação. Este modelo diminui o número de iterações para se obter uma solução ótima aproximada.

O segundo modelo de Kawato baseia-se em evidências de que áreas de associação sensorial do cortex (áreas 5 e 7) realizam a transformação de coordenadas e geração de comando motor através de algoritmo de aprendizado iterativo, isto é, por tentativa e erro. Neste caso, é necessário se conhecer a memória de curto prazo da história da trajetória e do torque do braço para se resolver o problema. O problema da dinâmica inversa é resolvido por uma rede neural de três camadas, treinada pelo algoritmo de retropropagação, ligada ao sistema em paralelo com o controlador do manipulador. Os ângulos desejados de trajetória são fornecidos simultaneamente ao controlador e à rede. Esta última produz um torque que somado ao torque do controlador é entregue ao robô. O torque do controlador é resultado da diferença entre o vetor ângulo desejado e aquele obtido de fato. Esta diferença cai para zero quando a rede completa o estágio de treinamento.

Atkeson e Reinkensmeyer [3] discutiram o uso de memória associativa em controle de robô. Eles propuseram um algoritmo do vizinho mais próximo paralelo que foi implementado numa *Connection Machine* [18]. Nesta abordagem, as experiências são armazenadas na memória e previsões são realizadas com base em buscas na memória. Foi empregado o modelo de memória associativa de conteúdo

enderessável - ACAM [25] para simulação de um braço planar com duas juntas. O emprego do vizinho mais próximo produziu resultados iniciais promissores.

Este modelo parte de uma trajetória pré-estabelecida e o robô usa os sinais de comando em conjunto com os sinais sensoriais realimentados para controlar o braço. O modelo armazena na memória, para cada instante de tempo t , os valores de posição, velocidade, aceleração angulares e torques de cada junta “ i ” $(\theta_i(t), \dot{\theta}_i(t), \ddot{\theta}_i(t), \tau_i(t))$.

O modelo da dinâmica inversa produz o torque necessário para se encontrar uma trajetória particular da seguinte maneira: $\tau_i(t) = \hat{R}^{-1}(\theta_i(t), \dot{\theta}_i(t), \ddot{\theta}_i(t))$. A transformação $\hat{R}^{-1}$ é representada neste modelo, por busca entre os valores armazenados na memória. Logo, dado um conjunto de valores $(\theta_i(t_d), \dot{\theta}_i(t_d), \ddot{\theta}_i(t_d))$ procura-se na memória o conjunto destas variáveis que seja mais próximo dos valores dados. O torque armazenado com o conjunto achado é assumido como o torque para as grandezas fornecidas. A busca exaustiva entre os vários componentes da memória é um fator limitante do método. Schaal e Atkeson [48] implementaram este esquema usando aprendizado fundamentado numa técnica de regressão não-paramétrica.

A resposta a mudanças em seu meio-ambiente deve ser rápida. Nesta direção, Lin [30] empregou aprendizagem por reforço, implementado em redes neurais, para criar estratégias de controle reativas sem qualquer conhecimento prévio dos efeitos causados pelas ações do robô. O agente recebe uma avaliação escalar do meio-ambiente que recompensa tarefas completadas com sucesso e penaliza tarefas que fracassam. O agente tem que maximizar os sinais que o premiam. O agente pode então generalizar para eventos imprevistos. Este método se mostrou mais eficiente que compilação de ações para melhoria de performance no domínio do mundo real. Em contrapartida, a convergência da aprendizagem foi tipicamente mais demorada e os planos produzidos nos primeiros estágios de aprendizagem eram perigosos para o robô.

4. MODELOS PROPOSTOS

A maioria dos modelos de redes neurais parcialmente recorrentes encontrados na literatura apresentam melhor desempenho quando utilizados para problemas de reconhecimento de seqüências temporais do que para problemas de produção de seqüências temporais. Espera-se obter um modelo de rede neural parcialmente recorrente que apresente bom desempenho em problemas de produção de seqüências temporais sem grande complexidade de treinamento.

Nesta dissertação, seis modelos de redes neurais parcialmente recorrentes são propostos para produzir seqüências temporais. Assim, se os pontos iniciais e finais de uma seqüência treinada são apresentados à qualquer uma das redes, os modelos deverão produzir a seqüência completa. Os modelos propostos empregam algoritmo de aprendizagem com ponto não-fixo.

Os seis modelos têm três camadas: uma camada de entrada, uma escondida e uma de saída. A estrutura de alimentação direta é a mesma para todos os modelos. A camada de entrada é totalmente conectada com a camada escondida que é totalmente conectada com a camada de saída.

O número de unidades nas camadas de entrada, escondida e de saída de qualquer arquitetura é constante. Para o problema discreto, o número de unidades nas camadas de entrada, escondida e de saída é igual a 16, 22 e 16 respectivamente. Para o problema contínuo é 15, 20 e 15 respectivamente. O número de unidades nas camadas de entrada e de saída é função da representação escolhida. O número de unidades na camada escondida foi estabelecido empiricamente. Este número foi o mínimo necessário para que a primeira arquitetura proposta resolvesse o problema discreto para o primeiro tipo de trajetória.

Figura 19: (a) Código de endereços para cada posição espacial permitida para os blocos; (b) Uma configuração espacial particular dos blocos.

A representação adotada para o problema discreto foi representar cada posição permitida no espaço para a movimentação dos blocos por combinações de 0's e 1's (Figura 19.a). Assim, a posição espacial de um bloco é representada por um conjunto de quatro unidades que assumem o endereço da posição do bloco como estado de ativação. A ativação de cada grupo de quatro unidades representa o endereço (posição espacial) de um bloco dentre as 12 possíveis posições. Por exemplo, se os quatro blocos são colocados como esboçado na Figura 19.b, então os blocos A, B, C, e D são representados por: 0 0 0 1, 0 0 1 1, 0 0 1 0, 0 0 0 0. respectivamente.

Em resumo, cada conjunto de posições espaciais dos blocos são representados por um vetor com dimensão 16 definido como declarado acima.

.As conexões de realimentação são de dois tipos: conexões de realimentação da camada de saída para a camada de entrada e conexões de realimentação local de saída. O primeiro grupo está presente nos seis modelos, por ser de fundamental importância em casos de produção de seqüências temporais, enquanto o segundo grupo existe em apenas quatro modelos. Conexões de realimentação local são conexões que ocorrem de uma unidade para ela mesma. Nesta dissertação, será utilizado uma extensão das realimentações locais, as conexões intracamadas, que ocorrem de uma unidade em uma camada para todas as outras unidades desta mesma camada, inclusive a própria unidade. Nos modelos estudados, estas conexões estão situadas na camada de entrada ou na camada escondida. As conexões de realimentação intercamadas são treináveis somente nas três últimas arquiteturas, enquanto as conexões intracamadas são todas treináveis. Todas as conexões de realimentação são atrasadas em um passo de tempo. Em resumo, as arquiteturas são propostas de acordo com a seguinte estratégia:

- A primeira arquitetura é tomada como referência. Ela emprega conexões de realimentação intercamadas para reproduzir seqüências temporais.
- As arquiteturas 2 e 3 são propostas para avaliar o efeito das conexões de realimentação intracamada na performance da primeira topologia.
- As arquiteturas 4, 5 e 6 transformam as conexões intercamadas um-para-um nos três primeiros modelos em conexões todas-para-todas, para avaliar a influência de se processar a informação realimentada de forma distribuída para todas as unidades de entrada no desempenho da rede.

Os seis modelos serão aplicados na resolução do problema de produção de seqüências temporais discretas, que consiste em planejar a trajetória do problema do mundo dos blocos. O modelo que apresentar melhor desempenho na resolução deste problema será aplicado para resolver o problema de produção de seqüências temporais contínuas, que consiste em planejar e controlar a trajetória do manipulador robô PUMA 560.

4.1. Parte comum a todos os modelos

Todos os modelos propostos apresentam a mesma estrutura de alimentação direta (Figura 20), consistindo de uma rede perceptron multicamadas (*Multilayer Perceptron* - MLP) com uma camada de entrada, uma camada intermediária e uma camada de saída. As regras de ativação, propagação e aprendizagem para esta parte são idênticas as da MLP treinadas pelo algoritmo de retropropagação. Além disso, os modelos usam aprendizagem *on-line* em conjunto com a estratégia professor forçante para a realimentação intercamada de saída.

Na fase de treinamento, as conexões de realimentação podem ser consideradas como uma extensão virtual da camada anterior à camada que está recebendo a realimentação. Estas unidades virtuais contêm uma cópia da saída das unidades da camada a ser realimentada no instante de tempo anterior. Por isto, as

arquiteturas podem ser treinadas utilizando o algoritmo de treinamento de retropropagação tradicional.

Figura 20: Estrutura de alimentação direta comum a todos os modelos.

O algoritmo de retropropagação [45] é o mais popular método para treinamento de redes neurais. A razão para sua popularidade é a simplicidade e o poder do algoritmo. O poder deriva do fato de que, diferente de seus precursores, a regra de aprendizagem perceptron e a regra de aprendizagem Wildrow-Hoff, o algoritmo de retropropagação pode ser empregado para treinar redes não-lineares de conectividade arbitrária. Tais redes são frequentemente requeridas para aplicações no mundo real. A idéia básica é antiga e simples: define-se uma função erro e usa-se o gradiente descendente para encontrar um conjunto de pesos que minimize o erro em uma tarefa particular. O algoritmo é tão simples que pode ser implementado em poucas linhas de código. Apesar da simplicidade, o algoritmo de retropropagação pode sofrer de convergência lenta e convergência para mínimos locais, entre outros problemas.

Considere um conjunto de treinamento que consiste de um conjunto de pares ordenados $[(\vec{r}, \vec{v})]$, onde $\vec{v}$ é um resultado que ocorre no contexto do evento $\vec{r}$. A meta da rede é aprender a relação entre $\vec{r}$ e $\vec{v}$. Pode-se imaginar que exista algum mapeamento desconhecido que relacione $\vec{r}$ e $\vec{v}$ e que esteja tentando encontrar uma boa aproximação para esta função. Redes neurais multicamadas são tipicamente não-

lineares, portanto redes neurais com alimentação direta fazem um tipo de regressão não-linear.

Para evitar mínimos locais e aumentar a velocidade do algoritmo de aprendizagem, fez-se uso neste trabalho de alguns métodos com taxa de aprendizagem variável e termo de *momentum*.

4.1.1. Algoritmo de aprendizagem

O conjunto de amostras utilizado para treinar os modelos consiste em um conjunto de seqüências. Cada seqüência representa uma trajetória e é formada por um conjunto de estados da rede. Portanto, cada par entrada-saída, representando um estado e seu sucessor que é apresentado para o treinamento das redes, é chamado de padrão. Cada conjunto de padrões que formam uma trajetória é chamado de seqüência. O conjunto total de seqüências é chamado de conjunto de treinamento.

O algoritmo de aprendizagem de retropropagação utilizado para treinar os seis modelos é descrito passo a passo a seguir [17]:

- Inicialização de pesos e unidades da rede:
 1. Os vários pesos são iniciados com valores reais e aleatórios entre -1 e 1.
 2. As unidades de processamento da camada de entrada, escondida e de saída são inicializadas com valor zero.
- Propagação dos padrões de entrada para a saída:
 3. Um padrão de entrada é apresentado às unidades de entrada da rede.

$$\{x_1, x_2, \dots, x_n\}$$

(4.1.1.1)

4. A propagação do sinal de entrada é dada por:

$$q_i^m = \sum_j \omega_{ij}^m x_j^m \quad (4.1.1.2)$$

onde q_i^m é a ativação da unidade i para a camada m sendo $m = 1 \dots M$; ω_{ij}^m é o valor do peso da conexão entre a unidade j de uma camada e a unidade i da camada anterior; x_j^m é o sinal de entrada para a unidade i saindo da unidade j .

5. A saída de cada unidade é dada por:

$$y_i^m = f(q_i^m) \quad (4.1.1.3)$$

onde $f(.)$ é uma função não-linear. Para o problema de produção de seqüências temporais discretas utilizou-se $g(.)$ como sendo a função sigmoidal, para o caso de produção de seqüências temporais contínuas, $f(.)$ é a função tangente hiperbólica.

6. Compara-se a saída obtida pela rede com a saída desejada. Calcula-se o erro médio quadrático por padrão entre a saída obtida e a saída desejada por:

$$E^p = \frac{1}{2} \sum_{i=1}^{n_s} (D_i^p - S_i^p)^2 \quad (4.1.1.4)$$

onde D_i^p e S_i^p são a saída desejada e a obtida, para a unidade i da camada de saída da rede quando apresentado o padrão p e n_s é o número de unidades da camada de saída.

7. Se o erro médio quadrático obtido for maior do que o erro pré-especificado, segue-se para o passo 10. Caso contrário, se não tiver chegado ao fim da seqüência, pega-se o próximo padrão desta seqüência como entrada para a rede e retorna-se para o passo 4. Se tiver chegado ao fim da seqüência vá para o passo 8.
8. Caso se tenha chegado ao fim da seqüência, propaga-se todos os padrões desta seqüência pela rede, calculando-se o erro para

cada padrão. Estes erros são somados e o erro obtido é chamado de erro da sequência. O erro da sequência é comparado com o erro desejado para a sequência, que é o erro pré-especificado multiplicado pelo número de padrões de cada sequência. Se o erro da sequência for maior do que o erro desejado, pega-se o primeiro padrão desta sequência como entrada para a rede e retorna-se ao passo 4. Caso o erro da sequência seja menor do que o desejado, e esta não seja a última do conjunto de sequências, pega-se o primeiro padrão da próxima sequência como entrada para a rede e retorna-se para o passo 4.

9. Caso a sequência seja a última do conjunto de treinamento, teremos completado uma época e então propaga-se todos os padrões de todas as sequências do conjunto de sequências pela rede calculando-se agora o erro para cada sequência. Estes erros são somados e o erro total obtido é chamado de erro do conjunto de sequências. O erro obtido do conjunto de sequências é comparado com o erro desejado. Este último é definido como o erro pré-especificado multiplicado pelo somatório do número de estados em todas as sequências. Se erro do conjunto de sequências for maior do que o erro desejado do conjunto de sequências, então pega-se o primeiro padrão da primeira sequência do conjunto de sequências como entrada para a rede e retorna-se ao passo 4. Caso o erro do conjunto de sequências seja menor do que o desejado, encerra-se o treinamento.

- Propagação do erro para trás.

10. Calcula-se os deltas para os pesos que chegam à camada de saída pela expressão:

$$\delta_i^M = f'(h_i^M)(D_i^M - S_i^M) \quad (4.1.1.5)$$

comparando-se a saída obtida S_i^M com a saída desejada D_i^M para o padrão p considerado.

11. Os delta das camadas anteriores são calculados pela equação:

$$\delta_i^{m-1} = f'(h_i^{m-1}) \sum_j \omega_{ji}^m \delta_j^m \quad (4.1.1.6)$$

para $m = M, M-1, \dots, 2$ até um delta ter sido calculado para todas as unidades.

12. A atualização dos pesos é dada por:

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \Delta \omega_{ij}(t) \quad (4.1.1.7)$$

onde

$$\Delta \omega_{ij}(t) = \eta(t) \delta_i^m y_j^m + \psi \Delta \omega_{ij}(t-1) \quad (4.1.1.8)$$

onde η é a taxa de aprendizagem e ψ o termo de *momentum*.

13. Apresenta-se novamente o mesmo padrão como entrada para a rede e retorna-se ao passo 4.

Os critérios de parada adotados foram um número de épocas máximo ou um valor máximo para o erro médio quadrático (4.1.1.4) para cada par entrada-saída de uma seqüência.

Não é fácil escolher valores apropriados para os parâmetros η e ψ para um problema particular [17]. Além disto, o melhor valor no começo do treinamento pode não ser o melhor no final do treinamento. Assim, o procedimento adotado foi atualizar os parâmetros automaticamente como parte do processo de aprendizagem. O processo consiste em verificar se a atualização atual dos pesos decresce a função de erro. Se não, η deve ser reduzido. Por outro lado, se vários passos em uma dada direção decrementa a função erro, então deve-se tentar incrementar η .

4.2. Modelo inicial

O primeiro modelo proposto (Figura 21) é uma rede parcialmente recorrente em que o sinal de cada unidade da camada de saída é realimentado para cada unidade da camada de entrada, através de conexões do tipo um-para-um.. Este modelo pode ser considerado um caso particular do modelo proposto por Jordan, fazendo-se a constante $\alpha = 0$ e também não é criado uma camada de contexto. O fato de não ser criado uma camada de contexto, torna a arquitetura mais simples, diminuindo a quantidade de pesos da arquitetura pois para o modelo proposto aqui, os pesos que ponderam a entrada da rede são os mesmo que ponderam a realimentação intercamada.

Espera-se que este modelo seja capaz de produzir seqüências simples e curtas por causa da realimentação entre camadas. Todos os resultados das outras topologias serão comparados com os resultados deste modelo, a fim de avaliar o papel desempenhado pelos diferentes tipos de recorrência em planejamento baseado em casos.

As conexões de realimentação da camada de saída para a camada de entrada neste modelo não são treináveis, têm seus pesos fixos e iguais a um. A função de ativação da i -ésima unidade de entrada é uma cópia da saída da rede defasada de um instante de tempo:

$$i_i(t+1) = o_i(t) \quad (4.2.1)$$

onde $i_i(t)$ e $o_i(t)$ são os estados de ativação da i -ésima unidade de entrada e de saída no tempo t .

No estágio de treinamento e durante a execução, o sinal de entrada para a rede é o ponto inicial e o ponto meta. Para este modelo contendo conexões de realimentação um-para-um, a entrada da rede no instante inicial é dada por:

$$i_i(0) = (is_i + gs_i)/2 \quad (4.2.2)$$

onde is_i e gs_i são os estados inicial e final da i -ésima unidade de entrada.

Esta foi uma forma encontrada para entrar com os pontos inicial e meta na mesma unidade de processamento, sendo cada um ponderado de 0.5.

Figura 21: Arquitetura de rede parcialmente recorrente proposta como referência.

4.3. Variações do modelo inicial

As outras cinco arquiteturas propostas nesta dissertação são derivadas do modelo acima. Primeiramente, são acrescentadas conexões de realimentação intracamadas no modelo tomado como referência. Acrescenta-se conexões de realimentação intracamada na camada de entrada e na camada escondida, obtendo-se assim dois novos modelos. Estas conexões intracamadas são do tipo todos-para-todos e treináveis. Os outros três modelos propostos são variações dos três primeiros, acrescentando-se agora conexões de realimentação todos-para-todos da camada de saída para a camada de entrada em cada um dos três modelos.

4.3.1. Incluindo realimentação local

Os dois primeiros modelos derivados do modelo tomado como referência possuem a mesma estrutura, acrescentando-se apenas conexões de realimentação intracamadas. Espera-se que tais conexões resolvam situações conflitantes entre

trajetórias que necessitem do histórico para atingir a meta desejada. A recorrência local processa o sinal de entrada a fim de considerar os estados passados.

Figura 22: Segundo modelo proposto no qual acrescenta-se realimentação intracamada na entrada.

O segundo modelo (Figura 22) tem um conjunto adicional de conexões de realimentação todos-para-todos da camada de entrada para ela mesma. A regra de ativação da i -ésima unidade de entrada é dada por:

$$i_i(t+1) = f\left(o_i(t) + \sum_{j=1}^{ni} k_{ij} i_j(t)\right) \quad (4.3.1.1)$$

onde $i_i(t)$ é o estado de ativação da i -ésima unidade de entrada no instante de tempo t ; k_{ij} são os pesos da camada de entrada; $o_i(t)$ é o estado de ativação da i -ésima unidade de saída no instante de tempo t e ni é o número de unidades na camada de entrada. A função $f(.)$ é a função de ativação de cada unidade de processamento.

A entrada da rede no instante inicial é dada pela equação 4.2.2

Figura 23: Terceiro modelo proposto no qual acrescenta-se realimentação intracamada na camada escondida.

A terceira arquitetura (Figura 23) tem conexões todos-para-todos da camada escondida para ela mesma. Esta opção adiciona inércia ao mapeamento realizado pela camada escondida. A regra de ativação da i -ésima unidade escondida é dada por:

$$h_i(t+1) = f\left(\sum_{j=1}^{ni} k_{ij} i_j(t) + \sum_{j=1}^{ne} \omega_{ij} h_j(t)\right) \quad (4.3.1.2)$$

onde h_j é o estado de ativação da j -ésima unidade escondida no instante de tempo t ; ω_{ij} são pesos da camada escondida e ne é o número de unidades na camada escondida.

A regra de ativação para as unidades de entrada segue as equações (4.2.1) e (4.2.2).

4.3.2. Realimentação Intercamada Distribuída

Os próximos três modelos possuem a mesma estrutura dos três primeiros modelos propostos, mudando apenas as conexões de realimentação da camada de saída para a camada de entrada de um-para-um para todos-para-todos. Esta modificação tem como objetivo diminuir as falhas da rede para recuperar seqüências

através do aumento da associação entre estados em um passo de tempo particular e o próximo passo de tempo.

Figura 24: Quarto modelo proposto: modelo referência no qual as conexões intercamadas são do tipo todos-para-todos.

O quarto modelo proposto (Figura 24) é uma modificação do modelo tomado como referência. O estado de ativação da i -ésima unidade de entrada para este modelo é dado por:

$$i_i(t+1) = f\left(\sum_{j=1}^{n_s} (k_{ij} o_j(t))\right) \quad (4.3.2.1)$$

onde $o_j(t)$ é o estado de ativação da j -ésima unidade de saída no passo de tempo t e n_s é o número de unidades na camada de saída.

A entrada da rede para o instante inicial é dada por:

$$i_i(0) = f\left(\sum_{j=1}^{n_s} (k_{ij} (is_i + ig_i) / 2)\right) \quad (4.3.2.2)$$

onde is_i e gs_i são os estados inicial e final da i -ésima unidade de entrada e k_{ij} são os pesos.

Figura 25: Quinto modelo proposto: semelhante ao modelo 2. Transforma-se conexões de realimentação da camada de saída para a camada de entrada de um-para-um para todos-para-todos.

Na arquitetura cinco (Figura 25), o estado de ativação da i -ésima unidade de entrada é dado por:

$$i_i(t+1) = f\left(\sum_{j=1}^{n_s} k_{ij} o_j(t) + \sum_{j=1}^{n_l} \omega_{ij} i_j(t)\right). \quad (4.3.2.3)$$

A ativação das unidades de entrada no instante inicial é dada por:

$$i_i(0) = f\left(\sum_{j=1}^{n_s} k_{ij} i s_j + \sum_{j=1}^{n_l} \omega_{ij} i g_j\right) \quad (4.3.2.4)$$

onde k e ω são pesos e $i s_j$ e $i g_j$ são os estados inicial e final da i -ésima unidade de entrada.

A sexta topologia (Figura 26) tem o estado de ativação da i -ésima unidade de entrada dado por (4.3.2.1) e o estado de ativação da j -ésima unidade escondida é dado por (4.3.1.2). A ativação das unidades de entrada no instante inicial é dada pela equação (4.3.2.2).

Figura 26: Sexto modelo proposto: semelhante ao modelo 3. Transforma-se conexões de realimentação da camada de saída para a camada de entrada de um-para-um para todos-para-todos.

4.4. Atuação dos modelos nos problemas discreto e contínuo

Para o problema do mundo dos blocos, o papel da rede neural será a do planejador (Figura 27), ou seja, a rede é treinada para gerar uma sequência de posições para os blocos, uma vez apresentada como entrada à rede a posição inicial e a posição meta para os blocos. As restrições dos problemas são absorvidas pela rede na fase de treinamento.

Figura 27: Diagrama de blocos para o problema de planejamento de trajetória do mundo dos blocos.

Para o problema de planejamento e controle de trajetória do braço mecânico, três problemas devem ser resolvidos: determinação da trajetória, cinemática inversa e

dinâmica inversa. Nesta dissertação, os três problemas serão tratados de uma só vez, ou seja, a rede deverá resolver os três problemas em uma única etapa como ilustrado na Figura 28.

Figura 28: Diagrama de blocos para o problema de planejamento e controle de trajetória do braço mecânico.

O próximo capítulo reporta os experimentos realizados com os modelos para verificar a função de cada tipo de realimentação e o desempenho dos modelos na solução dos problemas considerados nesta dissertação.

5. TESTES

Um dos objetivos desta dissertação é entender o papel das realimentações intracamadas e intercamadas em arquiteturas de redes neurais parcialmente recorrentes aplicadas à produção de seqüências temporais através de estudos empíricos do desempenho dos seis modelos propostos. Como descrito no Capítulo 4, um modelo apenas com realimentação intercamadas foi proposto e tomado como referência. Modificações específicas neste modelo foram feitas dando origem a cinco diferentes arquiteturas. As mudanças feitas no modelo original foram com relação à inserção de realimentação intracamada na camada de entrada e na camada escondida. Outra modificação foi quanto ao tipo de realimentação intercamada, podendo esta ser todos-para-todos ao invés de um-para-um.

Para se analisar o desempenho dos diferentes modelos diante dos problemas tratados nesta dissertação, alguns critérios de comparação foram estabelecidos. Para o problema do Mundo dos Blocos, os experimentos vão ser observados quanto à:

- Capacidade de recuperação de seqüências treinadas: Dados os pontos inicial e meta, verifica se a rede produz a seqüência de estados que a leva de um ponto a outro.
- Capacidade de generalização da rede: A capacidade de generalização tratada para este caso mede a capacidade que a rede apresenta para recuperar uma seqüência de estados, sendo dado como entrada o estado inicial e o estado meta. Este ponto inicial é um ponto intermediário das seqüências treinadas que não foi treinado anteriormente como ponto inicial.

Para o problema de planejamento e controle de trajetória do braço mecânico, os critérios adotados foram semelhantes ao caso anterior. Foi mantido o mesmo critério de capacidade de recuperação da sequência, porém modificou-se o critério de capacidade de generalização. A capacidade de generalização considerada terá três variações. A primeira será acrescentar ruído às posições iniciais treinadas e manter a posição final sem modificação para verificar se a rede consegue chegar ao ponto meta. Outro teste será acrescentar ruído ao ponto meta e manter o ponto inicial sem modificação. O último teste de capacidade de generalização considerado será acrescentar ruído tanto ao ponto inicial como ao ponto meta.

É importante salientar que, para o problema de planejamento e controle de trajetória do braço mecânico, não se tem por objetivo um estudo comparativo das arquiteturas, visto que será utilizado apenas a arquitetura que apresenta melhor desempenho em relação às outras no problema do mundo dos blocos. O objetivo será apenas verificar a capacidade do modelo em trabalhar com produção de sequências temporais para variáveis limitadas e contínuas. Para este problema, considera-se um acerto caso a arquitetura consiga atingir a meta, avaliando-se a trajetória obtida pela rede para atingir esta meta. Esta trajetória não precisa ser idêntica à desejada mas deve ser a mais próxima possível. Já para o problema do mundo dos blocos, mesmo que a arquitetura chegasse ao ponto meta, mas por uma sequência de pontos intermediários diferentes dos treinados, a trajetória seria considerada incorreta.

Em ambos os problemas, o critério de parada adotado no estágio de treinamento das redes são:

- o treinamento é encerrado quando todas as sequências são recuperadas com erro abaixo de um valor máximo ou;
- o treinamento acaba quando o estágio de treinamento atinge um número de épocas determinado sem convergir para o erro desejado.

Os programas deste trabalho foram escritos em linguagem C e implementados em um PC-Pentium.

5.1. Mundo dos Blocos

Para o problema do mundo dos blocos foram utilizados os seis modelos descritos anteriormente com a finalidade de recuperar seqüências treinadas e também de generalizar a aprendizagem.

Foram criados os quatro tipos de trajetórias mostrados na Figura 2, para treinar as arquiteturas. As trajetórias criadas não englobam todas as trajetórias possíveis no espaço de trabalho adotado, estas foram criadas para realizar uma tarefa específica. O conjunto de seqüências criadas para o treinamento das arquiteturas possuem em média 6, 7, 7 e 7 estados por seqüência para cada um dos quatro tipos de trajetórias respectivamente.

Para o primeiro tipo, onde se tem vários pontos iniciais e um único ponto meta (m-1), foram utilizadas noventa e uma trajetórias. Um exemplo de seqüência de estados para este tipo de trajetória é mostrado na Figura 29.

Figura 29: Seqüência de estados de uma trajetória do tipo $m-1$ criada para testar as arquiteturas.

Para o segundo tipo de trajetória, onde se tem apenas um ponto inicial e vários pontos metas (1-m), foram criadas quatro trajetórias distintas. Uma seqüência de estados para este tipo de trajetória utilizada para treinar a rede é ilustrada na Figura 30. É importante lembrar que neste caso não existem pontos em comum entre as trajetórias como na situação anterior.

Figura 30: Exemplo de sequência de estados de trajetória do tipo 1-m utilizada no treinamento da rede.

Para o terceiro tipo de trajetória em que se tem vários pontos iniciais e vários pontos metas onde as trajetórias não têm pontos em comum (m-m/sc), foram criados seis exemplos. A Figura 31 ilustra uma sequência de estados utilizada para treinar as arquiteturas.

Figura 31: Exemplo do tipo de trajetória m-m/sc corretamente recuperado.

Finalmente para o quarto tipo, onde se tem vários pontos iniciais e vários pontos finais para trajetórias que se cruzam (m-m/cc), foram criadas trinta trajetórias diferentes. As Figura 32 e Figura 33 ilustram duas sequências de estados com um ponto em comum.

Figura 32: Exemplo do tipo de trajetória m-m/cc corretamente calculada.

A diferença entre o número de seqüências temporais criadas para os diferentes tipos de trajetórias deve-se às restrições do problema. Para o primeiro tipo de trajetória, a restrição é de se ter trajetórias que saiam de pontos distintos e cheguem a um mesmo ponto, comum a todas as trajetórias, não importando se estas tenham pontos intermediários em comum. Isto permite criar um grande número de trajetórias no espaço de trabalho adotado. Para o segundo tipo de trajetória, as restrições são as seguintes: deseja-se sair de um mesmo ponto e chegar a vários pontos distintos no espaço sendo que as trajetórias não tenham pontos intermediários em comum. Isto dificulta a criação de um número grande de trajetórias pois o fato de não poder ter pontos intermediários em comum cria um gargalo no ponto seguinte ao inicial. Isto se deve ao fato de que a quantidade de posições permitidas distintas para sair de um mesmo ponto para um ponto seguinte é limitada. Para o terceiro tipo de trajetória, onde se tem como restrição sair de pontos distintos e chegar também a pontos distintos no espaço sendo que não hajam pontos em comum entre as trajetórias, o número de trajetórias possíveis é limitado pelo fato de não poder ter pontos em comum. Para o último tipo, em que se deseja sair de pontos distintos e chegar a pontos distintos no espaço permitindo um ponto em comum entre as trajetórias, possibilita um número consideravelmente maior de trajetórias.

Figura 33: Um exemplo do tipo de trajetória m-m/cc corretamente recuperada

É importante ressaltar que para cada tipo de trajetória foi criado apenas um grupo de trajetórias com as características especificadas, ou seja, para o caso onde se deseja sair de uma mesma posição para muitas posições meta, por exemplo, foi criado apenas um conjunto de trajetórias com estas características para o treinamento das arquiteturas em oposição à produção de todas as trajetórias possíveis para todos os pontos meta.

Criadas as trajetórias, treinou-se cada um dos seis modelos para recuperar cada um dos quatro tipos de trajetórias. Visando aumentar a velocidade de treinamento, as arquiteturas foram treinadas por partes, ou seja, o erro foi diminuído gradualmente. Primeiro treina-se a rede para um erro considerado alto, quando a rede convergir para este erro, diminui-se o erro máximo admissível e inicia-se o treinamento com os pesos obtidos na fase anterior e assim por diante até atingir um valor desejado.

O número de épocas para terminar cada estágio de treinamento é mostrado na Figura 34. Uma época considerada aqui é cada vez que todas as seqüências de padrões são apresentadas à rede. As trajetórias 2 e 3 são treinadas mais rapidamente em média porque há uma menor quantidade de seqüências.

Para a trajetória tipo m-1, observa-se que apenas a arquitetura 3 necessitou de um número maior de épocas, comparada com as outras arquiteturas, para convergir para um erro aceitável, permitindo-lhe recuperar todos os padrões de treinamento. A arquitetura 5 apresentou menor número de épocas dentre as seis. Para trajetórias 1-m,

as arquiteturas 2, 3 e 4 apresentaram número de épocas semelhantes, assim como as arquiteturas 1 e 6. Outra vez, a arquitetura 5 apresentou o menor número de épocas para convergir para um erro que lhe permitiu recuperar todos os padrões. Para a trajetória tipo m-m/sc as arquiteturas 1, 3, 4, 5 e 6 apresentaram número de épocas próximos. Apenas a arquitetura 2 necessitou de um número de épocas bem maior do que as outras cinco para recuperar todos os padrões de treinamento. Para a trajetória tipo m-m/cc, a arquitetura 5 convergiu para o menor erro dentre as seis arquiteturas, com o terceiro menor número de épocas, sendo esta a única arquitetura que conseguiu recuperar todos os padrões de treinamento.

Figura 34: Número de épocas de treinamento por trajetória para cada arquitetura.

É importante ressaltar que o número de épocas para a convergência da rede depende dentre outras coisas da inicialização dos pesos, que foram feitos de forma aleatória, da taxa de aprendizagem e do termo de momentum inicial. Isto porque se

os pesos forem inicializados em uma região próxima ao ponto mínimo desejado, e se a taxa de aprendizagem e o termo de momentum forem escolhidos adequadamente, a convergência é mais rápida. Se a taxa de aprendizagem e o termo de momentum forem muito grandes, a rede pode ficar oscilando em torno do ponto mínimo desejado, dificultando a convergência. Se a taxa de aprendizagem e o termo de momento forem muito pequenos, pode-se demorar muito para chegar ao ponto mínimo desejado.

Figura 35: Percentagem de seqüências corretamente recuperadas quando a entrada da rede são o estado inicial e o estado meta treinado.

Detalhes dos resultados obtidos para o modelo são apresentados nas tabelas do Apêndice II. Como pode ser observado na Tabela 1, que traz os resultados para o treinamento das seis arquiteturas, para o tipo de trajetória m-1, apenas a arquitetura 2 não conseguiu recuperar todos os padrões visto que não se conseguiu convergência para erro menor que 0.3. As arquiteturas 1, 4, 5 e 6 conseguiram recuperar todos os

padrões com o mesmo erro de 0.01. A arquitetura 3 conseguiu recuperar todos os padrões para um erro de 0.005. Para os tipos de trajetórias 1-m e m-m/sc (Tabela 2 e 3), todos os padrões foram recuperados por todas as arquiteturas, com o mesmo erro de 0.1. Para o tipo de trajetória m-m/cc (Tabela 4), considerado o de maior dificuldade, apenas a arquitetura 5 foi capaz de recuperar todos os padrões. A arquitetura 2 apresentou novamente o pior desempenho, não conseguindo convergência para erros menores do que 0.4. As arquiteturas 1, 3, 4 e 6 apresentaram desempenhos bastante similares. As arquiteturas 1 e 4 não convergiram para erros menores que 0.09 e as arquiteturas 3 e 6 não convergiram para erros menores que 0.08. A Figura 35 ilustra a percentagem de seqüências corretamente recuperadas pelos seis modelos quando os padrões apresentados como entrada para as redes são os pontos iniciais e finais treinados.

Uma seqüência é corretamente recuperada se todos os estados treinados são recuperados. Uma falha é caracterizada quando o estado meta não é alcançado ou se o estado meta é alcançado através de uma seqüência de estados distintos da seqüência de estados treinados. A Figura 36 ilustra uma seqüência de estados do quarto tipo de trajetória produzida pela arquitetura 1, onde a rede chega ao estado meta passando por estados não permitidos.

Figura 36: Seqüência de estados recuperados com falhas pela arquitetura 1 para a trajetória m-m/cc.

Para a avaliação da capacidade de generalização dos modelos, foi elaborado um conjunto de padrões de testes que consiste em partes das seqüências de estados

treinadas. Neste caso, não se utiliza o ponto inicial treinado como ponto inicial a ser apresentado aos modelos. Pontos treinados como intermediários passam a ser pontos iniciais. A primeira seqüência testada é iniciada no segundo ponto intermediário e termina no ponto meta treinado. A segunda seqüência testada começa no terceiro ponto intermediário treinado e acaba no ponto alvo. O procedimento é repetido para uma dada seqüência treinada até quando o ponto inicial de teste for o penúltimo ponto treinado da seqüência. Este procedimento é repetido para todas as seqüências treinadas. Apresenta-se as seqüências de teste às arquiteturas com os pesos ajustados para gerar os estados intermediários destas seqüências de testes. Os resultados do teste de generalização são apresentados na Figura 37.

Figura 37: Percentagem de seqüências recuperadas para o teste de generalização.

Como pode ser observado, todos os modelos apresentam uma baixa capacidade de generalização para este tipo de teste de generalização. Para a trajetória

do tipo m-1, as arquiteturas 1 e 3 apresentaram melhor desempenho em relação às outras quatro arquiteturas. A arquitetura 2 e 6 apresentaram o pior desempenho. Para a trajetória do tipo 1-m, a arquitetura 5 mostrou-se bem melhor do que as outras cinco arquiteturas; já as arquiteturas 2 e 4 tiveram o pior desempenho. Para a trajetória do tipo m-m/sc, a arquitetura 5 apresentou novamente o melhor desempenho, enquanto as arquiteturas 4 e 6 obtiveram o pior desempenho. Para a trajetória do tipo m-m/cc, a arquitetura 1 mostrou-se melhor do que as outras cinco, enquanto as arquiteturas 2 e 4 apresentaram o pior desempenho.

5.1.1 Discussão dos resultados

Com base nos resultados anteriormente apresentados, deseja-se destacar nesta seção, características para cada um dos modelos estudados que os tornem apropriados para resolver problemas de produção de seqüências temporais.

Analisando primeiramente a capacidade das arquiteturas de recuperar seqüências treinadas, cujos resultados são apresentados na Figura 35, observa-se que a arquitetura 2 obteve o pior desempenho dentre os seis modelos estudados, mostrando-se não apropriada para este tipo de tarefa. Isto pode ser justificado pela influência das realimentações locais que são significativamente mais relevantes do que as realimentações intercamada. As arquiteturas 1 e 4 mostraram boa performance em reproduzir os três primeiros tipos de trajetórias, não tendo desempenho tão bom quando aplicadas ao tipo de trajetória m-m/cc. Isto sugere que apenas conexões de realimentação intercamada não são suficientes para capacitar a rede a identificar seqüências em situação de conflito, ou seja, seqüências que possuam ponto em comum com outras seqüências. Isto sugere também, que o fato das conexões de realimentação intercamadas serem um-para-um ou distribuídas não aumenta significativamente a capacidade de aprendizagem da rede quando não se inclui conexões intracamadas. As arquiteturas 1, 3 e 6 apresentam performance semelhantes sugerindo que realimentação local na camada escondida não exerce grande influência na aprendizagem de seqüências com pontos em comum. Na maioria dos casos, estas novas ligações apenas tornaram o treinamento mais longo. Isto mostra a importância

da escolha da posição da realimentação intracamada na resolução de problemas. A arquitetura 5 apresentou melhor desempenho na reprodução das seqüências treinadas, visto que foi a única que conseguiu recuperar todas as seqüências de padrões para os quatro tipos de trajetórias estudados. A combinação de conexões de realimentação intercamadas com conexões intracamadas parece aumentar o poder da rede para reproduzir seqüências.

Analisando a Figura 37, observa-se que as arquiteturas mostram uma baixa capacidade de generalização. Isto sugere que se um dado estado inicial é significativamente diferente dos estados iniciais treinados (um ponto intermediário da seqüência), os modelos não são capazes de recuperar as seqüências treinadas. As arquiteturas 1 e 3, que apresentam conexões de realimentação da camada de saída para a camada de entrada um-para-um, superaram em todos os casos as arquiteturas 4 e 6 que apresentam conexões de realimentação da camada de saída para a camada de entrada todos-para-todos. Isto sugere que quando não se tem conexões de realimentação intracamada ou quando estas estão na camada escondida, conexões de realimentação da camada de saída para a camada de entrada um-para-um são mais apropriadas do que todos-para-todos, quando o interesse é a capacidade de generalização do modelo. Esta situação aparentemente ocorre pois nas topologias 4 e 6 existe maior amarração entre estados sucessivos e não existe uma estrutura mais apropriada para identificar as seqüências. Pode-se observar que quanto maior o grau de dificuldade da trajetória, menor a capacidade de generalização dos modelos. A arquitetura 5 mostrou em média o melhor desempenho em relação aos outros modelos

5.2 Trajetória de um braço mecânico

A aplicação dos modelos estudados nesta dissertação para o problema do mundo dos blocos foi uma forma de quantizar os resultados obtidos para o desempenho destes modelos permitindo compará-los e escolher o que melhor se adapte à resolução de problemas de produção de seqüências temporais.

Para o problema de planejamento e controle de trajetória do braço mecânico foi utilizado apenas a arquitetura que apresentou melhor desempenho na resolução do problema anterior. Deseja-se realizar a tarefa relativa ao robô como maneira de viabilizar este tipo de arquitetura para resolver problemas de produção de seqüências temporais contínuas, tais como planejamento e controle de trajetórias de braços mecânicos.

A arquitetura terá como entrada o ponto inicial e o ponto meta em coordenadas cartesianas no espaço tridimensional (x, y, z), os seis ângulos e os seis torques para as juntas do braço mecânico relacionados aos pontos dados. Ela deverá gerar toda a trajetória para ir do ponto inicial ao ponto meta. Para cada estado intermediário gerado pela rede, esta deve apresentar, além do ponto em coordenadas cartesianas no espaço tridimensional, os seis ângulos e os seis torques relacionados a cada junta do manipulador. Portanto, a rede terá 15 variáveis de entrada e 15 variáveis de saída. Com isto, a rede resolve os problemas de cinemática inversa e dinâmica inversa simultaneamente, que são dois problemas de extrema complexidade dentro de robótica clássica.

Pela análise dos resultados apresentados na seção anterior, observa-se que a arquitetura que apresentou melhor desempenho foi a arquitetura 5 (Figura 25) ou seja, a arquitetura com realimentação intercamadas todos-para-todos e realimentação intracamadas na camada de entrada também todos-para-todos.

Para gerar os quatro tipos de trajetórias (Figura 2) a serem utilizadas no treinamento, utilizou-se a ferramenta *Robotics* do Matlab. Este *software* gera a trajetória para ir de um ponto a outro. Basta estipular o número de pontos intermediários, o ponto inicial e o ponto final. Através da cinemática inversa, este *software* gera os ângulos do braço mecânico para cada posição espacial das trajetórias criadas e também os torques em cada atuador do braço, através da dinâmica inversa.

Para cada tipo de trajetória foram criados três caminhos distintos, cada um com dez pontos intermediários. A Figura 38 ilustra as três trajetórias criadas para treinar a rede para os tipos de trajetórias m-1 e 1-m. Para o tipo de trajetória m-1, considera-se caminhando no sentido de vários pontos iniciais para um único ponto

meta, ou seja, os pontos indicados na figura por B serão os pontos iniciais enquanto o ponto A é o ponto meta. Para o tipo de trajetória 1-m, considera-se o sentido inverso.

Figura 38: Trajetórias do tipo 1-m e m-1, criadas para o treinamento do braço mecânico.

A Figura 39 ilustra as três trajetórias criadas para o treinamento da rede para o tipo de trajetória m-m/sc. Estas consistem de trajetórias partindo de pontos distintos e chegando a pontos diferentes sem ponto em comum entre as seqüências de estados.

Figura 39: Trajetórias criadas para o tipo de trajetória m-m/sc. O ponto inicial é representado por I e o ponto meta é representado por M .

A Figura 40 ilustra as três trajetórias do tipo m-m/cc utilizadas para treinar a arquitetura escolhida. Estas possuem pontos de cruzamento entre si.

Uma vez geradas as trajetórias, obtém-se os conjuntos de ângulos para as juntas do braço mecânico para cada posição espacial através da cinemática inversa. De posse do conjunto de ângulos, calcula-se os torques em cada ponto da trajetória através de técnicas de robótica clássica para cálculo da dinâmica inversa. Com isso, obtém-se todo o conjunto de dados necessários para o treinamento da rede. A entrada para a rede será a posição espacial, os ângulos das juntas e os torques.

Figura 40: Trajetórias do tipo m-m/cc criadas para treinar a rede. O ponto inicial é representado por *I* e o ponto meta é representado por *M*.

A função de transferência utilizada nas unidades de processamento da arquitetura foi a função tangente hiperbólica $f(h)=\tanh(h)$. Por este motivo, os dados para o treinamento da rede devem ser normalizados para o intervalo de -1 e 1. O processo de normalização adotado foi dividir todos os valores de cada variável de entrada pelo maior valor absoluto encontrado para esta variável em todo o conjunto de treinamento para cada um dos quatro tipos de trajetórias, ou seja, tomando-se como exemplo a variável x , encontra-se o maior valor para esta variável, considerando-se as três trajetórias tratadas para cada um dos quatro tipos de trajetórias estudados nesta dissertação. Divide-se todos os valores desta variável pelo

maior valor encontrado adicionado de um termo de folga, obtendo-se assim todos os valores para cada variável entre -1 e 1. O termo de folga utilizado foi 20% do maior valor da variável portanto, o valor normalizado é calculado segundo a equação:

$$\text{valor}_{\text{normalizado}} = \frac{\text{valor}}{1,2 * \text{maior}} \quad (5.3.1)$$

Este termo de folga foi incluído para tornar possível a inserção de ruído durante o teste de generalização, caso não houvesse uma folga o maior valor seria 1 e quando acrescentasse ruído este valor seria maior que 1.

A arquitetura utilizada na resolução deste problema obteve bom desempenho para o teste de produção de trajetória. A Figura 41 ilustra as três trajetórias desejadas e geradas pela rede para o tipo m-1 sendo apresentados como entrada para a rede os pontos espaciais x, y e z, os seis ângulos e os seis torques, um para cada junta do braço mecânico.

Figura 41: Trajetórias desejada e gerada pela arquitetura para o tipo m-1. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

As trajetórias geradas pela rede e as desejadas estão sobrepostas, portanto a rede consegue recuperar a trajetória treinada com boa precisão. O modelo mostra-se eficiente para o planejamento de trajetória do tipo m-1.

Os ângulos das juntas gerados pela rede e os desejados para as três trajetórias do tipo m-1 são apresentados na Figura 42.

Figura 42: Ângulos do braço mecânico para as trajetórias do tipo m-1. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

Como pode ser observado, o modelo recupera muito bem os ângulos de cada junta do braço mecânico para todas as três trajetórias do tipo m-1.

A Figura 43 ilustra os torques obtidos pela rede e os torques desejados nos atuadores das juntas do braço mecânico para as trajetórias do tipo m-1.

Figura 43: Torques desejados e gerados pela rede nos atuadores das juntas do braço mecânico para as trajetórias do tipo m-1. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

Os torques de cada junta do braço mecânico também foram corretamente recuperados para as trajetórias do tipo 1-m. O modelo utilizado nesta tarefa apresentou bom desempenho, produzindo as trajetórias e gerando os ângulos e os torques das juntas do robô para as três trajetórias do tipo m-1, de forma precisa.

Para as trajetórias do tipo 1-m a rede também conseguiu recuperar todas as trajetórias treinadas com grande precisão. A Figura 44 apresenta as três trajetórias

geradas pela rede com suas respectivas trajetórias desejadas para este tipo de trajetória.

Figura 44: Trajetórias obtidas pela rede e desejada para o tipo 1-m. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

Os ângulos e os torques das juntas, para o tipo de trajetória 1-m, também foram recuperados com boa precisão. Os gráficos que ilustram estes resultados estão no Apêndice III.

As trajetórias recuperadas pela arquitetura e as trajetórias desejadas para as três trajetórias do tipo m-m/sc estão ilustradas na Figura 45.

Figura 45: Trajetórias geradas pela rede e desejadas para o tipo m-m/sc. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

O modelo mostra bom desempenho na produção das trajetórias do tipo m-m/sc produzindo todas as trajetórias treinadas.

Os ângulos e os torques gerados pela arquitetura e os desejados para as seis juntas do manipulador para a trajetória do tipo m-m/sc também foram recuperadas com boa precisão e os resultados estão ilustrados no Apêndice III.

Figura 46: Trajetórias geradas pela rede e desejadas para o tipo m-m/cc. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

A Figura 46 mostra as trajetórias geradas pela rede e as trajetórias desejadas para o tipo m-m/cc. Este tipo de trajetória é considerado o de maior grau de complexidade com relação aos outros três tipos, pois as trajetórias apresentam pontos de conflitos, ou seja, pontos em que duas trajetórias se cruzam e a rede deve ser capaz de seguir a trajetória correta.

O modelo abordado, recuperou a primeira (Figura 46.a) e a terceira (Figura 46.c) trajetórias de forma bastante precisas. Para a segunda trajetória (Figura 46.b), apresentou um erro com ordem de grandeza de 10^{-2} , mostrando bom desempenho também para este tipo de trajetória, pois conseguiu optar pela trajetória correta em todos os pontos de cruzamento.

Os ângulos desejados e os gerados pela rede, das seis juntas do manipulador, para o tipo de trajetória m-m/cc, são apresentados na Figura 47.

A arquitetura gera todos os ângulos para as juntas de braço mecânico corretamente.

Figura 47: Ângulos desejados e gerados pela rede, das juntas do manipulador para a trajetória tipo m-m/cc. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

Os torques gerados pela rede e os desejados para os atuadores das seis juntas do manipulador são ilustrados na Figura 48, para o tipo de trajetória m-m/cc

Figura 48: Torques gerados pela rede e desejados para os seis atuadores das juntas do manipulador para a trajetória tipo m-m/cc. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

Como pode ser observado pelos gráficos, a rede é capaz de gerar com grande precisão as trajetórias treinadas, e também gerar os ângulos e torques para as juntas do robô em uma única etapa, mostrando-se viável na resolução de problemas de

produção de seqüências temporais. Isto sugere que o modelo proposto nesta dissertação, é capaz de resolver problemas de grande complexidade de forma bastante precisa como o problema de planejamento e controle de trajetória de braço mecânico.

O teste de capacidade de generalização para este problema foi realizado de três formas: primeiramente acrescenta-se ruído apenas no ponto inicial, apresentado como entrada à rede, e verifica-se se esta é capaz de chegar ao ponto meta seguindo a trajetória desejada. Outro teste de capacidade de generalização foi acrescentar ruído apenas ao ponto meta, apresentado como entrada à rede, mantendo o ponto inicial sem modificação e verifica-se se a rede é capaz de atingir o ponto meta seguindo a trajetória desejada. O último teste de capacidade de generalização aplicado ao modelo, foi acrescentar ruído aos pontos inicial e meta apresentados à rede como entrada à rede e verifica-se se esta é capaz de seguir a trajetória desejada até o ponto meta. Para os três testes de capacidade de generalização, o ruído acrescentado aos pontos inicial e meta foi de 5% do valor da variável, ou seja, para a variável x do ponto inicial é adicionado 5% do seu valor. Neste documento será apresentado apenas os resultados do teste mais crítico, no caso, quando é acrescentado ruído aos pontos inicial e meta da trajetória.

Figura 49: Trajetórias obtidas pela rede e desejadas para o tipo 1-m com inserção de ruído de 5% para os pontos inicial e meta. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

A Figura 49 ilustra as trajetórias desejadas e obtidas pela rede para as três trajetórias do tipo 1-m, acrescentando-se 5% de ruído aos ponto inicial e meta que são apresentados como entrada à rede.

Para a primeira trajetória do tipo 1-m (Figura 49.a) verifica-se que o modelo gera a trajetória de forma satisfatória, divergindo de forma considerável, apenas para o último ponto. Para a segunda trajetória do tipo 1-m (Figura 49.b) a trajetória é recuperada com pequenas divergências entre os pontos gerados e desejados. A terceira trajetória para este tipo (Figura 49.c) é que apresentou maior divergência entre os pontos desejados e obtidos pela rede.

Figura 50: Trajetórias desejadas e geradas pela arquitetura para o tipo m-1 com inserção de ruído de 5% para os pontos inicial e meta. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

A Figura 50 ilustra as trajetórias desejadas e obtidas pelo modelo para as três trajetórias do tipo m-1 acrescentando-se 5% de ruído aos pontos inicial e meta apresentados como entrada para a rede.

O modelo conseguiu recuperar todas as trajetórias para o tipo m-1, com pequena diferença entre os pontos desejados e obtidos para as três trajetórias.

A Figura 51 ilustra as trajetórias desejadas e geradas pela rede para as três trajetórias do tipo m-m/cc tendo sido inserido ruído de 5% ao ponto inicial e final apresentados como entrada para a rede..

Figura 51: Trajetórias desejadas e geradas pela arquitetura para o tipo m-m/cc com inserção de ruído de 5% para os pontos inicial e meta. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

Para o tipo de trajetória m-m/cc, que é o mais complexo, o modelo não apresenta bom desempenho para gerar as trajetórias, quando aplicado ruído à sua entrada. No início da trajetória o modelo recuperou os pontos com erro pequeno, mas este erro aumenta com a distância do ponto inicial.

Figura 52: Trajetórias desejadas e geradas pela arquitetura para o tipo m-m/sc com inserção de ruído de 5% para os pontos inicial e meta. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

A Figura 52 ilustra os resultados obtidos para as três trajetórias do tipo m-m/sc, quando inserido ruído de 5% ao ponto inicial e meta apresentados à rede como entrada.

O modelo recuperou a primeira trajetória do tipo m-m/sc (Figura 52.a) com pequena diferença entre os pontos desejados e obtidos, da trajetória. Para a segunda

trajetória (Figura 52.b), também recuperou bem, divergindo de forma considerável do ponto desejado, apenas no último ponto. Para a terceira trajetória (Figura 52.c), o modelo não conseguiu recuperar bem.

Os gráficos ilustrando os ângulos e torques de cada junta do manipulador, para os quatro tipos de trajetórias, encontram-se no Apêndice III.

5.2.1 Discussão dos resultados

Com base nos resultados obtidos para os testes de capacidade de produção de seqüências temporais e de capacidade de generalização, aplicados ao modelo que apresentou melhor desempenho na resolução do problema discreto, pode-se verificar que o modelo é capaz de realizar rastreamento (tracking).

O teste de produção de seqüências temporais mostrou que o modelo é capaz de produzir as trajetórias treinadas, apresentando bom desempenho em todos os casos estudados. O modelo apresenta também, capacidade de trabalhar com mais de uma trajetória, sendo utilizado neste trabalho três trajetórias para cada tipo de trajetória. Os resultados mostram que o modelo é capaz de produzir trajetórias, ângulos e torques em uma única etapa.

Pelo teste de generalização, verifica-se que o modelo consegue recuperar satisfatoriamente as trajetórias do tipo $m-1$, quando acrescenta-se ruído ao sistema, enquanto para as trajetórias $1-m$, em geral, a rede produz apenas o ponto final com erro considerável. Isto pode ser explicado, pelo fato de que para o tipo $m-1$, o ponto meta é um forte atrator, não permitindo que o último ponto da trajetória divirja. Verifica-se também, que o ruído tem maior influência em trajetórias com pontos de conflitos. O modelo tem suave degradação com a inclusão do ruído

O teste de generalização sugere que o modelo perde a capacidade de realizar rastreamento com o aumento do ruído e com o aumento da complexidade do problema.

6. CONCLUSÕES

Redes neurais recorrentes são de extrema importância pois são capazes de modelar sistemas dinâmicos não-lineares. Tais sistemas são bastante complexos no campo de controle. Embora seja de muita importância, ainda não se é capaz de dizer quais os tipos de redes neurais recorrentes são mais apropriadas para determinadas tarefas. Redes neurais completamente recorrentes apresentam algumas características não desejáveis como: dificuldade de treinamento e lenta convergência. Redes neurais parcialmente recorrentes apresentam características interessantes uma vez que em muitas tarefas apresentam desempenho igual ou melhor do que as redes completamente recorrentes e apresentam um grau de complexidade de treinamento menor do que as redes completamente recorrentes. A maioria dos estudos sobre redes neurais parcialmente recorrentes relacionadas com problemas de seqüências temporais utilizam-nas para reconhecimento de seqüências temporais. Nesta dissertação, foram propostos e estudados seis modelos de redes neurais parcialmente recorrentes aplicados na resolução de problemas de produção de seqüências temporais.

Inicialmente, os modelos foram aplicados na resolução de um problema discreto de produção de seqüências temporais. Tal problema consiste em produzir uma seqüência de estados que levem um conjunto de quatro blocos de um estado inicial a um estado meta dentro de um espaço de trabalho com doze posições possíveis, que estão divididas em três colunas com quatro posições cada. Este espaço de trabalho apresenta algumas restrições que determinam posições permitidas para o movimento dos blocos. Tal problema foi escolhido com o intuito de quantizar de forma clara os resultados obtidos pelos diferentes modelos, permitindo assim compará-los e classificá-los quanto à viabilidade de tais modelos na resolução de problemas de produção de seqüências temporais, mesmo que de forma empírica.

O modelo que apresentou melhor desempenho na resolução do problema de produção de seqüências temporais discretas foi aplicado na resolução de um problema de produção de seqüências temporais contínua, buscando generalizar a viabilidade do referido modelo de rede neural parcialmente recorrente para resolver problemas de produção de seqüências temporais.

O problema de produção de seqüência temporal contínuo tratado nesta dissertação consiste em planejamento e controle de trajetória de um braço mecânico. Tal problema é de grande importância e complexidade. A rede resolve os problemas de geração de trajetória, cinemática inversa e dinâmica inversa simultaneamente.

Os testes realizados com os seis modelos no primeiro problema e com o modelo que apresentou melhor desempenho para o segundo problema foram:

- Teste de capacidade de produção de seqüências temporais.
- Teste de capacidade de generalização de seqüências temporais.

Para o problema do mundo dos blocos, as trajetórias do tipo 1-m e m-m/sc apresentam treinamento mais rápido do que as do tipo m-1 e m-m/cc pois são em menor quantidade de padrões de treinamento. No entanto, a relação entre o número de épocas para treinamento não é diretamente proporcional ao número de seqüências para cada tipo de trajetória. As simulações sugerem que este número de épocas é proporcional à complexidade da tarefa

Em média, a arquitetura 5 foi a que apresentou menor número de épocas para ser treinada. Ela também atingiu, em média, os menores valores de erro durante o treinamento. Isto sugere que a combinação de realimentação inter e intracamadas, realizada na arquitetura 5, apresenta melhor capacidade de aprender entre aquelas que foram implementadas.

A arquitetura que apresentou melhor desempenho para este problema foi a arquitetura 5, com realimentação da camada de saída para a camada de entrada todos-para-todos e realimentação intracamada na camada de entrada todos-para-todos. Este modelo foi o único que conseguiu recuperar todos os padrões treinados para os quatro tipos diferentes de trajetórias, criadas para o treinamento das arquiteturas. De

forma geral, apresentou também melhor capacidade de generalização, embora esta tenha sido baixa. Devido ao seu bom desempenho na resolução deste problema, esta arquitetura foi a escolhida para ser aplicada ao problema de produção de seqüências temporais contínuas.

O modelo aplicado ao problema de planejamento e controle de trajetória do braço mecânico apresentou um bom desempenho para produção das trajetórias do braço mecânico, conseguindo reproduzir as trajetórias treinadas com grande precisão.

Para os casos de menor complexidade, verifica-se o modelo é capaz de produzir seqüências próximas as desejadas, mesmo na presença de ruído. A rede perde sua capacidade de rastreamento com o aumento do ruído e da complexidade do problema.

Uma limitação do modelo é quanto à quantidade de trajetórias que consegue recuperar. Para o problema de produção de seqüências temporais contínuas, trabalha-se com um erro muito menor que o problema discreto, isto é, no último caso qualquer valor maior que 0.5 é aproximado para 1 e qualquer valor menor ou igual a 0.5 é aproximado para 0, logo a quantidade de seqüências que o modelo consegue mapear é limitada.

Uma grande contribuição do modelo proposto é quanto à capacidade de gerar a trajetória tendo como saída os pontos espaciais, os ângulos e os torques nas juntas para cada estado intermediário da trajetória. Ou seja, a rede executa os problemas de geração de trajetória, cinemática inversa e dinâmica inversa em uma única etapa.

Um dos modelos propostos, a arquitetura 2, mostrou-se completamente inviável para produzir seqüências temporais visto o desempenho desta nos teste realizados.

A análise dos resultados obtidos para o problema do mundo dos blocos para os seis modelos estudados sugere que conexões de realimentação intercamada capacita a rede a identificar seqüências, enquanto que as conexões de realimentação intracamadas são mais apropriadas para fornecer informações do passado permitindo à rede levar em conta seu histórico em situações presentes. A combinação de ambas conexões de realimentação possibilita à rede produzir seqüências temporais com bom desempenho.

Uma sugestão para aumentar a capacidade de armazenar informação e facilitar o treinamento das redes para o problema de planejamento e controle de trajetória seria separar cada problema e resolver cada um utilizando uma rede com as mesmas características da arquitetura 5 proposta, ou seja, treinar uma rede para realizar o problema de planejamento de trajetória gerando apenas os pontos espaciais. Treinar outra rede para resolver o problema de cinemática inversa, ou seja, para encontrar os ângulos para as juntas do manipulador sendo dadas as posições espaciais e, finalmente, treinar outra rede para executar o problema de dinâmica inversa. Isto porque tais problemas podem apresentar dificuldades diferentes e a divisão de um problema maior em subproblemas pode facilitar a resolução deste.

Outra sugestão para trabalhos futuros seria inserir conhecimento preliminar para a rede.

7. REFERÊNCIAS BIBLIOGRÁFICAS

- [1] Almeida, L. B., (1987). "A learning rule for asynchronous perceptrons with feedback in a combinatorial environment", in *Proceedings IEEE 1st. International Conference on Neural Network*, 609-618.
- [2] Armstrong, B., Khatib, O., & Burdick, J., (1986). "The explicit dynamic model and inertial parameters of the PUMA 560 arm", in *Proc. IEEE Int. Conf Robotics and Automation*, vol. 1, (San Francisco, USA), 510-518.
- [3] Atkeson, C. G. & Reinkensmeyer, D. J. (1991). "Using associative content-addressable memories to control robots", in W. T. Miller III, R. S. Sutton and P. J. Werbos (eds). *Neural Networks for Control*, 255-285. Cambridge, MA: MIT Press.
- [4] Back, A. D. e Tsoi, A. C., (1991). "FIR and IIR synapses, a new neural network architecture for time series modelling", *Neural Computation*, vol.3, no.3, pp.375-385.
- [5] Corke, P. I. & Armstrong-Hélouvry, B., (1994). "A search for consensus among model parameters reported for PUMA 560 Robot", *IEEE Conf. On Robotics and Automation*, 1608-1613.
- [6] Corke P. I., (1996). "A robotics toolbox for MATLAB", *IEEE Robotics and Automation Magazine*, vol 3, nº 1, 24-32.
- [7] Craig J. J., (1989). *Introduction to robotics: mechanics and control*, 2ª ed. Addison-Wesley.
- [8] Denavit, J. & Hartenberg, R. S., (1955). "A kinematic notation for lower-pair mechanisms based on matrices", *Journal of Applied Mechanics*, 215-221
- [9] Elman, J. L. (1990). "Finding structure in time". *Cognitive Science*, vol. 14, 179-211.
- [10] Erol, K., Nau, D. S. e Subrahmanian, V. S, (1995). "Complexity, decidability and undecidability results for domain-independent planning", *Artificial Intelligence*, vol. 76, 75-88.
- [11] Fahlman, S. E., (1989). "Faster learning variations on backpropagation: An empirical study", *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann Publishers, 38-50.

-
- [12] Frasconi, P., Gori, M. e Soda, G. (1992). "Local feedback multilayered networks", *Neural Computation*, vol.4, 120-130.
- [13] Fu, K. S., Gonzalez, R. C., & Lee, C.S.G. (1987). *Robotics: Control, Sensing, Vision and Intelligence*. New York: McGraw-Hill Book Company.
- [14] Gupta, M. M., Rao, D. H., (1994). "Neuro-control systems: A tutorial", in *Neuro-Control Systems: Theory and Applications*, editado por Madan M. Gupta e Dandina H. Rao, IEEE Press.
- [15] Hecht-Nielsen, R. (1987). "Kolmogorov's mapping neural network existence theorem", in *Proceedings of the International Conference on Neural Networks (San Diego, 1987)*, vol. 3, 11-14, IEEE, New York.
- [16] Hecht-Nielsen, R. (1990). *Neurocomputing*, Addison-Wesley, Reading, MA.
- [17] Hertz, J., Krogh, A. e Palmer R. G. (1991), *Introduction to the Theory of Neural Computing*, Addison-Wesley.
- [18] Hillis, D. (1985). *The Connection Machine*. Cambridge, MA: MIT Press.
- [19] Hinton, G. E., (1989). "Deterministic Boltzmann learning performs steepest descent in weight-space", *Neural Computation*, vol. 1, no. 1, 143-150.
- [20] Jordan, M. I., (1989). "Serial order: A parallel distributed approach", in *Advances in Connectionist Theory: Speech*, J. L. Elman and D. E. Rumelhart (eds.), Erlbaum.
- [21] Josin, G. (1988). "Neural-space generalization of a topological transformation". *Biological Cybernetics*, vol. 59, 283-290.
- [22] Kawato, M. (1989). "Adaptation and learning in control of voluntary movement by the central nervous system". *Advanced Robotics*, vol. 3, n° 3.
- [23] Kawato, M. (1991). "Computational schemes and neural network models for formation and control of multijoint arm trajectory", in W. T. Miller III, R. S. Sutton & P. J. Werbos (eds). *Neural Networks for Control*, 197-228. Cambridge, MA: MIT Press.
- [24] Kohonen, T. (1989). *Self-Organization and Associative Memory*. Berlin, Germany: Springer-Verlag. Third Edition.
- [25] Kohonen, T. (1980). *Content-Addressable Memories*. New York: Springer-Verlag.

-
- [26] Koivo, A. J. (1989). *Fundamentals for Control of Robotic Manipulator*, John Wiley & Sons, Inc.
- [27] Kolmogorov, A. N. (1957). "On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition", *Doklady Akademii Nauk USSR*, 114, 679-681.
- [28] Kremer, S.C., (1995). "On the computational power of Elman-style recurrent networks", *IEEE Transactions on Neural Networks*, vol. 6 n° 4, 1000 -1004.
- [29] Kröse, B. J. A. e van der Smagt, P. P. (1993). *An Introduction to Neural Networks*, Handouts, quinta edição, University of Amsterdam.
- [30] Lin, L. J. (1993). "Reinforcement learning for robots using neural networks", *PhD Thesis*, Carnegie Mellon University.
- [31] Lippmann, R. P. (1987). "An introduction to computing with neural nets", *IEEE Magazine on Acoustics, Signal, and Speech Processing*, vol. 4, 4-22.
- [32] Ludik, J. , Prins, W., Meert, K. e Catfolis, T. (1997). "A comparative study of fully and partially recurrent networks", *International Conference on Neural Networks*, vol. 1, 292-297.
- [33] Marr, D. (1982). *Vision*. New York: W. H. Freeman.
- [34] Martinez, T. M., Ritter, H. J., & Schulten, K. J. (1990). "Three-dimensional neural net for learning visuomotor coordination of a robot arm", *IEEE Transactions on Neural Networks*, vol. 1, 131-136.
- [35] McAllester, D. e Rosenblitt, D., (1991). "Systematic nonlinear planning", *Ninth National Conference on Artificial Intelligence. (AAAI-91)*, vol. 2, 634-639.
- [36] McCulloch , W. S. e Pitts, W., (1943). "A logical calculus of ideas immanent in nervous activity." *Bull Math Biophysics*, vol. 5, 115-133.
- [37] McDermott, D. e Hendler J., (1995). "Planning: What it is, what it could be. An introduction to the special issue on Planning and Scheduling", *Artificial Intelligence*, vol. 76, 1-16.
- [38] Minsky, M. L. e Papert, S. A., (1990). *Perceptrons*. Cambridge: MIT Press. Expanded edition.
- [39] Mozer, M. C., (1989). "A focused back-propagation algorithm for temporal pattern recognition", *Complex Systems*, vol. 3, 349-381.

- [40] Nerrand, O., Roussel-Ragot, P., Urbani, D., Personnaz, L., e Dreyfus, G., (1994). "Training recurrent neural networks: Why and how? An illustration in dynamical process modeling", in *IEEE Transactions on Neural Networks*, vol. 5, nº2.
- [41] Pearlmutter, B. A. (1995). "Gradient calculations for dynamic recurrent neural networks: A survey", *IEEE Transactions on Neural Networks*, vol. 6, nº 5, 1212-1228.
- [42] Pineda, F., (1987). "Generalization of backpropagation to recurrent neural networks", *Physical Review Letters*, vol. 19, nº.59, 2229-2332.
- [43] Psaltis, D., Sideris, A., & Yamamura, A. A. (1988). "A multilayer neural network controller", *IEEE Control Systems Magazine*, vol. 8, 17-21.
- [44] Ritter, H. J., Martinez, T. M., & Schulten, K. J. (1992). *Neural Computation and Self-Organizing Maps: An Introduction*. Reading: Addison-Wesley.
- [45] Rumelhart, D. E., Durbin R., Golden, R. e Chauvin Yves (1995). "Backpropagation: The basic theory", in *Backpropagation: Theory, Architectures, and Applications*, editado por Yves Chauvin e David E. Rumelhart, Lawrence Erlbaum Associates.
- [46] Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986). "Learning internal representations by error propagation", in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, D. E. Rumelhart, J. L. McClelland, and the PDP Research Group (eds.), MIT Press, Cambridge, Mass.
- [47] Rumelhart, D. E. & McClelland, J. L., (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, The MIT Press.
- [48] Schaal, S. & Atkeson, C. G. (1994). Robot juggling: implementation of memory-based learning. *IEEE Control Systems Magazine*, vol. 14, nº 1, 57-71.
- [49] Sejnowski, T. J. e Rosenberg, C. R. (1987). "Parallel networks that learn to pronounce English text", *Complex Systems*, vol. 1, 145-168.
- [50] Spreecher, D. A. (1993). "A universal mapping for Kolmogorov's superposition theorem", *Neural Networks*, vol. 6, nº 8, 1089-1094.
- [51] Stornetta, W. S., Hogg, T. e Huberman, B. A., (1988). "A dynamic approach to temporal pattern processing", in *Neural Information Processing Systems*, D. Z. Anderson (Ed.), 750-759.
- [52] Tsoi, A. C. e Back, A. D., (1994). "Locally recurrent and globally feedforward networks: A critical review of architectures", *IEEE Transactions on Neural Networks*, vol. 5, no.2, 229-239.

-
- [53] van der Smagt, P., Groen, F. e Kröse, B. (1993). "Robot hand-eye coordination using neural networks", *Technical Report (CS-93-10) of the Department of Computer Systems*, University of Amsterdam.
- [54] Wada, Y. & Kawato, M. (1993). "Neural network model for arm trajectory formation using forward and inverse dynamics models", *Neural Networks*, vol. 6, n° 7, 919-932.
- [55] Williams, R. J. e Zipser, D. (1989a). "A learning algorithm for continually running fully recurrent neural networks", *Neural Computation*, vol. 1, 270-280.
- [56] Williams, R. J. e Zipser, D. (1989b). "Experimental analysis of the real-time recurrent learning algorithm", *Connection Science*, vol. 1, 87-111.
- [57] Williams, R. J., e Zipser, D. (1990). "Gradient-based learning algorithms for recurrent connectionist networks", *Technical Report NU-CCS-90-9*, College of Computer Science, Northeastern University, Boston.
- [58] Zipser, D. (1989). "A subgrouping strategy that reduces complexity and speeds up learning in recurrent networks", *Neural Computation*, vol. 1, pp. 552-558.

APÊNDICE 1

Parâmetros de cinemática e dinâmica referentes ao manipulador robô PUMA 560 utilizado nesta dissertação.

Tabela 1: Constantes cinemáticas, α em graus, a e d estão em m.

I	α_i	a_i	d_i
1	90	0	0
2	0	0.4318	0
3	-90	0.0203	0.15005
4	90	0	0.4318
5	-90	0	0
6	0	0	0

Tabela 2: Valores da massa (kg) e centro de gravidade (m) dos elos.

i	m_i	rx_i	ry_i	rz_i
1	0	0	0	0
2	17.4	-0.3638	0.006	0.2275
3	4.8	-0.0203	-0.0141	0.070
4	0.82	0	0.019	0
5	0.34	0	0	0
6	0.09	0	0	0.032

Tabela 3: Momentos de inércia sobre centro de gravidade (kg-m²).

I	I_{xx_i}	I_{yy_i}	I_{zz_i}	I_{xy_i}	I_{yz_i}	I_{xz_i}
1	0	0.35	0	0	0	0
2	0.13	0.524	0.539	0	0	0
3	0.066	0.086	0.0125	0	0	0
4	1.8e-3	1.3e-3	1.8e-3	0	0	0
5	0.3e-3	0.4e-3	0.3e-3	0	0	0
6	0.15e-3	0.15e-3	0.04e-3	0	0	0

Tabela 4: Inércia induzida (ARMATURE) e prazer suficiente

i	Jm_i	G_i
1	200e-6	-62.6111
2	200e-6	107.815
3	200e-6	-53.7063
4	33e-6	76.0364
5	33e-6	71.923
6	33e-6	76.686

APÊNDICE 2

Tabelas com os resultados dos treinamentos das seis arquiteturas para cada um dos quatro tipos de trajetórias. As Tabelas de 5 a 8 apresentam os resultados da fase de teste de produção de seqüências temporais. As Tabelas de 9 a 12 apresentam os resultados da fase de teste de capacidade de generalização para os 6 modelos estudados.

Tabela 5: Resultados obtidos no treinamento dos seis modelos, para o teste de produção de seqüências temporais, para a trajetória em que se tem vários pontos iniciais e um único ponto final.

	erro	n° de padrões não recuperados	n° de padrões recuperados	n° total de padrões	n° épocas
Arquitetura 1	0.1	63	28	91	60
	0.05	23	68	91	112
	0.02	5	86	91	173
	0.01	0	91	91	238
Arquitetura 2	0.5	91	0	91	176
	0.3	88	3	91	232
Arquitetura 3	0.1	49	42	91	126
	0.05	18	73	91	234
	0.02	4	87	91	370
	0.01	1	90	91	492
	0.005	0	91	91	772
Arquitetura 4	0.1	51	40	91	51
	0.05	18	73	91	87
	0.02	5	86	91	148
	0.01	0	91	91	196
Arquitetura 5	0.1	40	51	91	62
	0.05	18	73	91	103
	0.02	3	88	91	136
	0.01	0	91	91	165
Arquitetura 6	0.1	43	48	91	142
	0.02	2	89	91	186
	0.01	0	91	91	252

Tabela 6: Resultados obtidos no treinamento das seis arquiteturas, para o teste de produção de seqüências temporais, no caso em que se tem um único ponto inicial e vários pontos metas, sem pontos de cruzamento entre as trajetórias.

	erro	n° de padrões não recuperados	n° de padrões recuperados	n° total de padrões	n° épocas
Arquitetura 1	0.1	0	4	4	42
Arquitetura 2	0.1	0	4	4	32
Arquitetura 3	0.1	0	4	4	35
Arquitetura 4	0.1	0	4	4	32
Arquitetura 5	0.1	0	4	4	16
Arquitetura 6	0.1	0	4	4	40

Tabela 7: Resultados obtidos no treinamento das seis arquiteturas, para o teste de produção de seqüências temporais, no caso em que se tem vários pontos iniciais e vários pontos metas, sem pontos de cruzamento entre as trajetórias.

	erro	n° de padrões não recuperados	n° de padrões recuperados	n° total de padrões	n° épocas
Arquitetura 1	0.1	0	6	6	30
Arquitetura 2	0.1	0	6	6	79
Arquitetura 3	0.1	0	6	6	31
Arquitetura 4	0.1	0	6	6	32
Arquitetura 5	0.1	0	6	6	36
Arquitetura 6	0.1	0	6	6	38

Tabela 8: Resultados obtidos no treinamento das seis arquiteturas, para o teste de produção de seqüências temporais, quando se tem vários pontos iniciais e vários pontos metas com cruzamento entre as trajetórias.

	erro	n° de padrões não recuperados	n° de padrões recuperados	n° total de padrões	n° épocas
Arquitetura 1	0.1	12	18	30	461
	0.09	7	23	30	808
Arquitetura 2	0.8	30	0	30	36
	0.5	30	0	30	294
	0.4	30	0	30	757
Arquitetura 3	0.1	11	19	30	782
	0.08	10	20	30	945
Arquitetura 4	0.1	10	20	30	390
	0.09	9	21	30	647
Arquitetura 5	0.1	9	21	30	230
	0.05	4	26	30	336
	0.03	2	28	30	456
	0.02	1	29	30	587
	0.015	0	30	30	651
Arquitetura 6	0.1	11	19	30	37
	0.08	8	22	30	148

Tabela 9: Resultados obtidos no treinamento dos seis modelos, para o teste de generalização, para a trajetória em que se tem vários pontos iniciais e um único ponto final.

	n° de padrões não recuperados	n° de padrões recuperados	n° total de padrões	porcentagem de padrões recuperados
Arquitetura 1	276	120	396	30,3%
Arquitetura 2	396	0	396	0%
Arquitetura 3	274	122	396	30,8%
Arquitetura 4	327	69	396	17,42%
Arquitetura 5	304	92	396	23,23%
Arquitetura 6	396	0	396	0%

Tabela 10: Resultados obtidos no treinamento das seis arquiteturas, para o teste de generalização, no caso em que se tem um único ponto inicial e vários pontos metas, sem pontos de cruzamento entre as trajetórias.

	n° de padrões não recuperados	n° de padrões recuperados	n° total de padrões	porcentagem de padrões recuperados
Arquitetura 1	17	1	18	5,6%
Arquitetura 2	18	0	18	0%
Arquitetura 3	16	2	18	11,1%
Arquitetura 4	18	0	18	0%
Arquitetura 5	13	5	18	27,8%
Arquitetura 6	16	2	18	11,1%

Tabela 11: Resultados obtidos no treinamento das seis arquiteturas, para o teste de generalização, no caso em que se tem vários pontos iniciais e vários pontos metas, sem pontos de cruzamento entre as trajetórias.

	n° de padrões não recuperados	n° de padrões recuperados	n° total de padrões	porcentagem de padrões recuperados
Arquitetura 1	29	1	30	3,3%
Arquitetura 2	28	2	30	6,67%
Arquitetura 3	29	1	30	3,3%
Arquitetura 4	30	0	30	0%
Arquitetura 5	26	4	30	13,3%
Arquitetura 6	30	0	30	0%

Tabela 12: Resultados obtidos no treinamento das seis arquiteturas, para o teste de generalização, no caso em que se tem vários pontos iniciais e vários pontos metas, com pontos de cruzamento entre as trajetórias.

	n° de padrões não recuperados	n° de padrões recuperados	n° total de padrões	porcentagem de padrões recuperados
Arquitetura 1	157	4	161	2,48%
Arquitetura 2	161	0	161	0%
Arquitetura 3	158	3	161	1,86%
Arquitetura 4	161	0	161	0%
Arquitetura 5	159	2	161	1,24%
Arquitetura 6	160	1	161	0,62%

APÊNDICE 3

Ângulos e torque produzidos pelo modelo utilizado na resolução do problema de planejamento e controle de trajetória do braço mecânico.

Figura 1: $\hat{\theta}$ Ângulos desejados e gerados pela rede, das juntas do manipulador, para a trajetória tipo 1-m. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

Figura 2: Torques desejados e gerados pela rede, das juntas do manipulador, para a trajetória tipo 1-m. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

Figura 3: Ângulos desejados e gerados pela rede, das juntas do manipulador, para a trajetória tipo m-m/sc. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

*b34WEⁿμέ p_{TT}8

Figura 4: Torques desejados e gerados pela rede, das juntas do manipulador, para a trajetória tipo m-m/sc. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

Figura 5: Ângulos desejados e gerados pela rede, das juntas do manipulador para a trajetória tipo 1-m, com inserção de ruído de 5%. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

Figura 6: Torques desejados e gerados pela rede, das juntas do manipulador para a trajetória tipo 1-m, com inserção de ruído de 5%. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

Figura 7: Ângulos desejados e gerados pela rede, das juntas do manipulador para a trajetória tipo m-1, com inserção de ruído de 5%. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

Figura 8: Torques desejados e gerados pela rede, das juntas do manipulador para a trajetória tipo m-1, com inserção de ruído de 5%. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

Figura 9: Ângulos desejados e gerados pela rede, das juntas do manipulador para a trajetória tipo m-m/cc, com inserção de ruído de 5%. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

Figura 10: Torques desejados e gerados pela rede, das juntas do manipulador para a trajetória tipo m-m/cc, com inserção de ruído de 5%. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

Figura 11: Ângulos desejados e gerados pela rede, das juntas do manipulador para a trajetória tipo m-m/sc, com inserção de ruído de 5%. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.

Figura 12: Torques desejados e gerados pela rede, das juntas do manipulador para a trajetória tipo m-m/sc, com inserção de ruído de 5%. Os pontos representados por círculos são as trajetórias desejadas e os pontos representados por x são as trajetórias produzidas pela rede.