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

MAURICIO MENEGAZ

Aplicação da rede GTSOM para navegação de robôs móveis utilizando aprendizado por reforço

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

Prof. Dr. Paulo M. Engel
Orientador

Porto Alegre, março de 2009

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Menegaz, Mauricio

Aplicação da rede GTSOM para navegação de robôs móveis utilizando aprendizado por reforço / Mauricio Menegaz. – Porto Alegre: PPGC da UFRGS, 2009.

62 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2009. Orientador: Paulo M. Engel.

1. Robótica. 2. Redes neurais artificiais. 3. Aprendizado por reforço. I. Engel, Paulo M.. II. Título.

“It has become appallingly obvious that
our technology has exceeded our humanity.”
— ALBERT EINSTEIN

AGRADECIMENTOS

Agradeço aos meus pais, sem eles nada disso teria sido possível. À minha esposa, que sempre me deu apoio nos momentos difíceis. Às minhas irmãs, pela companhia e amizade. Aos meus amigos. Ao meu orientador, pela ajuda no desenvolvimento deste trabalho.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

GTSOM	Growing Temporal Self Organizing Map
PDM	Processo de Decisão de Markov
SOM	Self Organizing Map
TDSOM	Time Delayed Self-Organizing Map
TKM	Temporal Kohonen Map
CSOM	Contextual Self Organizing Map
GCS	Growing Cell Structures
IGG	Incremental Grid Growing
GG	Growing Grid
GNG	Growing Neural Gas
IA	Inteligência Artificial

LISTA DE SÍMBOLOS

$\overline{\mathbf{x}}(t)$	Sinal interno da memória diferenciadora-integradora
$\mathbf{x}(t)$	Sinal de entrada da memória diferenciadora-integradora
$\mathbf{y}(t)$	Sinal de saída da memória diferenciadora-integradora
μ	Constante da memória diferenciadora-integradora
η	Constante da memória diferenciadora-integradora
$\mathbf{w}^h(t)$	Valor da sinapse da unidade de habituação
α	Taxa de recuperação da unidade de habituação
β	Constante de velocidade da unidade de habituação
γ	Constante de variação da unidade de habituação
$\mathbf{S}$	Estímulo externo aplicado à unidade de habituação
$\mathbf{z}(t)$	Entrada da rede GTSOM (incluindo o conteúdo da memória)
$\mathbf{w}_i(t)$	Vetor de pesos do neurônio i da rede GTSOM
a	Constante de decaimento do estímulo externo da unidade de habituação
v	Neurônio vencedor da rede GTSOM
$e_v(t)$	Taxa de aprendizado do neurônio vencedor
$e_n(t)$	Taxa de aprendizado dos neurônios vizinhos do vencedor
ρ	Parâmetro de plasticidade da rede GTSOM
δ	Parâmetro de novidade da rede GTSOM
I_{max}	Idade máxima das conexões da rede GTSOM
s_t	Estado do ambiente no tempo t
a_t	Ação escolhida pelo agente no tempo t
r_{t+1}	Recompensa recebida pela ação executada em t
π_t	Política de seleção de ações no tempo t
R_t	Retorno total esperado pelo agente
γ	Taxa de desconto dos retornos futuros
$P^a_{ss'}$	Probabilidade de transição do estado s para o estado s' ao executar a ação a

$R^a_{ss'}$	Reforço recebido ao passar do estado s para o estado s' executando a ação a
$V^\pi(s)$	Valor do estado s na política de ações π
$Q^\pi(s,a)$	Valor da ação a no estado s, para a política π

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Neste trabalho será descrita uma arquitetura de agente robótico autônomo projetada para ser capaz de criar uma representação de estado do ambiente e de realizar o aprendizado de tarefas simples em cima desta representação.

A rede GTSOM (BASTOS, 2007) foi selecionada como método para classificação de estados. Sua tarefa é transformar os dados multidimensionais e contínuos lidos dos sensores em uma representação discreta, permitindo o uso de aprendizado por reforço convencional. Algumas modificações no algoritmo da rede foram necessárias para que pudesse ser aplicada neste contexto. Juntamente com esta rede, foi utilizado um mapa de grade que permite associar as experiências sensoriais com sua localização espacial.

Enquanto a rede GTSOM é o ponto central de um sistema de classificação de estados, o algoritmo Q-Learning de aprendizado por reforço foi utilizado para a realização da tarefa. Utilizando a representação compacta de estado criada pela rede auto-organizável, o agente aprende as ações que devem ser executadas em cada ponto, para atingimento de seus objetivos.

O modelo foi testado com um experimento que consiste em encontrar um objeto em um labirinto. Os resultados obtidos nos testes mostraram que o modelo consegue segmentar adequadamente o espaço de estados, e realiza o aprendizado da tarefa. O agente consegue aprender a evitar colisões e memorizar a localização do alvo, podendo chegar até ele independentemente de sua posição inicial. Além disso, é capaz de expandir sua representação sempre que se depara com situações não conhecidas, ao mesmo tempo que gradualmente remove da memória estados associados a experiências que não se repetem.

Palavras-chave: Robótica, redes neurais artificiais, aprendizado por reforço.

Using the GTSOM network for mobile robot navigation with reinforcement learning

ABSTRACT

This work describes an architecture for an autonomous robotic agent that is capable of creating a state representation of its environment and learning how to execute simple tasks using this representation.

The GTSOM Neural Network was chosen as the method for state clustering. It is used to transform the multidimensional and continuous state signal into a discrete representation, allowing the use of conventional reinforcement learning techniques. Some modifications on the algorithm were necessary so that it could be used in this project. This network is used together with a grid map algorithm that allows the model to associate the sensor readings with the places where they ocurred.

While the GTSOM network is the main component of a state clustering system, the Q-Learning reinforcement learning method was chosen for the task execution. Using the compact state representation created by the self-organizing network, the agent learns which actions to execute at each state in order to achieve its objectives.

The model was tested in an experiment that consists in finding the path in a maze. The results show that it can divide the state space in an useful way, and is capable of executing the task. It learns to avoid collisions and remembers the location of the target, even when the robot's initial position is changed. Furthermore, the representation is expanded when the agent faces an unknown situation, and at the same time, states associated with old experiences are forgotten.

Keywords: robotics, neural networks, reinforcement learning.

1 INTRODUÇÃO

Este capítulo inicia com a contextualização e motivação do trabalho, feita na seção 1.1. Em seguida, serão definidos os objetivos do estudo e suas contribuições relevantes. Também será informada a estrutura geral do texto, na seção 1.4

1.1 Contexto e motivação do estudo

O presente trabalho tem como foco principal aplicações na área de robótica. Problemas típicos desta área envolvem a capacidade de processar em tempo real dados sensoriais multi-dimensionais, com grande redundância e dependência temporal.

Quando o robô não possui de antemão um mapa do local em que se encontra, ele deve construir uma memória dos lugares visitados à medida que vai explorando. Este é o problema conhecido como auto-localização. Nele, a informação lida a partir dos sensores é utilizada para dois fins: o mapeamento, que é a representação do local que está sendo explorado, e a localização, que consiste em determinar em qual ponto do mapa o robô se encontra. Para auxiliar nesta tarefa, utiliza-se também a informação das distâncias percorridas, que é combinada com os dados sensoriais utilizando métodos para modelagem e filtragem do erro de medida, como o filtro de Kalman.

Embora algumas tarefas simples possam ser executadas com comportamento puramente reativo, sem necessidade de manter uma memória, atividades mais complexas exigem que o agente tenha noção do espaço onde se encontra. Além disso, ele deve ser capaz de memorizar a localização de objetos de interesse, e dirigir-se a estes locais quando julgar necessário para o alcance de seus objetivos. Usa-se o termo mapa cognitivo para denotar esta representação do espaço.

A tarefa que deve ser executada influencia diretamente na escolha do tipo de mapa a ser utilizado. Neste trabalho, a tarefa de interesse consiste em encontrar um objeto em um ambiente desconhecido. Considera-se que a localização do objeto não muda, mas o agente deve ser capaz de encontrá-lo independentemente de sua posição inicial. Desta forma, não basta apenas memorizar o caminho a ser seguido. É necessário ter conhecimento do ambiente para localizar-se, e a partir daí determinar qual a direção que deve ser tomada.

A rede GTSOM (BASTOS, 2007) foi desenvolvida visando aplicações na área de robótica. Suas características construtivista e de auto-organização a tornam adequada à utilização no problema do processamento sensorial. A rede é capaz de modificar sua topologia, adicionando novos nodos e arestas à medida que novas situações vão aparecendo, e removendo estruturas não utilizadas. Isto é necessário porque, como o ambiente de operação não é conhecido inicialmente, o agente vai construindo uma representação de forma incremental.

Porém, para realizar a tarefa não basta apenas construir um mapa. É necessário apren-

der qual a sequência de ações que devem ser tomadas para atingir o objetivo, a partir da posição atual. Para isto, utilizou-se neste trabalho o método de aprendizado por reforço. A rede GTSOM constitui a base de um modelo para identificação do estado do ambiente, em cima do qual o agente aprende as ações que devem ser executadas.

Ainda que haja trabalhos relatando a utilização de redes neurais auto-organizáveis na tarefa de identificação de agrupamentos do espaço sensorial, esta proposta distingue-se das demais devido à natureza da rede utilizada, que reúne as características de modificação dinâmica da estrutura e representação da dependência temporal dos dados. Também destaca-se a forma com que tal representação é integrada ao método de aprendizado.

1.2 Objetivos

O objetivo deste trabalho é projetar uma arquitetura de agente autônomo capaz de criar uma representação do ambiente, utilizando a rede GTSOM como um identificador de categorias no espaço sensorial, e realizar o aprendizado de tarefas simples através do método de aprendizado por reforço em cima desta representação. O agente deve ser capaz de aprender de modo on-line (sem distinção entre fase de treinamento e operação) e construir simultaneamente o mapa perceptivo e a função de valor de estados.

Para alcançar este objetivo, foram definidas as seguintes metas intermediárias:

- Verificar a viabilidade da aplicação da rede GTSOM na tarefa de identificação de agrupamentos no espaço sensorial, e realizar adaptações no algoritmo de construção da rede, conforme necessário;

- Desenvolver um simulador que permita testar a operação do agente e verificar os resultados obtidos;

- Propor uma arquitetura de máquina inteligente, baseada no ferramental estudado, para executar tarefas de exploração do ambiente

- Testar a utilização da arquitetura projetada na resolução de tarefas simples, utilizando o método de aprendizado por reforço (Q-Learning);

- Analisar a possibilidade de implementação da arquitetura em robôs reais.

1.3 Contribuições relevantes

Este trabalho está alinhado com um conjunto de esforços desenvolvidos pelo grupo de pesquisa no qual se insere. Dentre suas principais contribuições, podemos destacar:

- Adaptações na rede GTSOM visando tornar a representação criada mais estável, de forma que seus nodos possam ser interpretados como estados de um PDM (Processo de Decisão de Markov). Estas adaptações, bem como os experimentos que mostraram serem necessárias, serão apresentados no capítulo 5.

- Proposta de um modelo de agente que cria uma representação do ambiente associando dados sensoriais, espaciais e ações executadas. O agente foi testado com sucesso em uma tarefa de aprendizado de trajetória em um labirinto simples, como será mostrado no capítulos 6 e 7.

1.4 Estrutura do texto

O capítulo 2 apresenta os fundamentos teóricos das redes neurais auto-organizáveis e introduz a rede GTSOM, que foi utilizada no desenvolvimento deste trabalho. Em seguida, no capítulo 3, são mostradas as formas mais comuns de representação de ambientes: mapas de grade, mapas topológicos e os mapas híbridos, que combinam características dos dois primeiros. O capítulo 4 descreve os conceitos de aprendizado por reforço, e também apresenta aplicações desta forma de aprendizado que se assemelham ao modelo aqui proposto. No capítulo 5 inicia-se o desenvolvimento deste trabalho. São apresentados alguns experimentos iniciais realizados com a rede GTSOM, e propostas adaptações visando uma melhor adequação da rede ao problema estudado. O modelo proposto utilizando a versão adaptada da rede GTSOM juntamente com outros componentes é descrito em detalhes no capítulo 6. Logo após, são apresentados os resultados obtidos, no capítulo 7, e as conclusões e considerações finais, no capítulo 8.

2 REDES NEURAIS AUTO-ORGANIZÁVEIS

Neste capítulo serão abordadas as redes neurais conhecidas como mapas auto-organizáveis. Tal tipo de rede é baseado no processo de aprendizado competitivo. Os neurônios da camada de saída competem entre si para serem ativados ou disparados, com o resultado de que apenas um deles está ligado em cada instante de tempo. Uma forma de induzir este tipo de competição é utilizando conexões laterais inibitórias entre os neurônios. Outra maneira de atingir o mesmo resultado é através de um juiz externo, que determina qual o neurônio vencedor.

Em um mapa auto-organizável, os neurônios estão colocados em nós de uma grade que é normalmente uni ou bidimensional. Ao longo do processo de aprendizado, os neurônios se tornam seletivamente associados a padrões de entrada. As localizações dos neurônios se tornam ordenadas de forma que um sistema de coordenadas significativo no espaço de características é criado sobre a grade.

2.1 A rede SOM (Self Organizing Map)

A rede SOM, também chamada de mapa de Kohonen (KOHONEN, 1982), tem o objetivo de transformar um padrão de sinal incidente de dimensão arbitrária em um mapa discreto uni ou bidimensional, realizando esta transformação adaptativamente e de uma maneira topologicamente ordenada. O mapa consiste em uma grade n-dimensional de neurônios. No caso em que n = 1, ela equivale a uma linha, e quando n = 2, um retângulo. As dimensões da grade devem ser determinadas *a priori* pelo projetista. Cada um dos neurônios está conectado a todos os nós fonte da camada de entrada, através de um vetor de pesos $\mathbf{w}$. Ao longo desta seção, a dimensão do espaço de entrada será denotada por m, e o número total de neurônios na grade, l.

O algoritmo de treinamento da rede inicia atribuindo a todos os pesos da rede valores aleatórios. Uma vez inicializados, há 3 processos que governam a formação do mapa, que são (HAYKIN, 2001):

- Competição: para cada padrão de entrada, os neurônios da grade calculam seus respectivos valores de uma função discriminante. O neurônio com maior valor da função discriminante é considerado o vencedor. Normalmente, o critério utilizado é a menor distância euclidiana. Quando os vetores de entrada são normalizados, este critério equivale a maximizar o produto interno do vetor de pesos com o padrão de entrada. Se $\mathbf{x}$ é um padrão de entrada, então o índice $i(\mathbf{x})$ do neurônio que melhor casa com ele é dado pela equação 2.1.

$$i(\mathbf{x}) = \arg\min_j ||\mathbf{x} - \mathbf{w}_j||,\ j = 1, 2..l \tag{2.1}$$

- Cooperação: o neurônio vencedor determina a localização de uma vizinhança topológica de neurônios excitados, fornecendo assim a base para cooperação com os neurônios vizinhos. De um ponto de vista neurobiológico, há evidências de que um neurônio que está disparando tende a excitar mais fortemente os neurônios de sua vizinhança imediata do que aqueles distantes dele. Portanto, deve-se escolher uma função de vizinhança $h_{j,i}$ que possua esta característica. Uma função normalmente utilizada é a gaussiana (equação 2.2).

$$h_{j,i} = \exp\left(-\frac{d_{j,i}^2}{2\sigma^2}\right) \tag{2.2}$$

  onde d_{ji} é a distância euclidiana entre os neurônios i e j, e σ é o desvio padrão, que regula a largura da vizinhança. Para uma melhor convergência do algoritmo, é recomendável que a largura σ da função de vizinhança diminua com o tempo.

- Adaptação Sináptica: Consiste em ajustar os pesos dos neurônios excitados de maneira que a sua resposta ao padrão de entrada seja melhorada. Para cada neurônio j, é feito o ajuste conforme a equação 2.3.

$$\mathbf{w}_j(n+1) = \mathbf{w}_j(n) + \eta(n)h_{j,i}(n)(\mathbf{x}(n) - \mathbf{w}_j(n)) \tag{2.3}$$

  onde i é o índice do neurônio vencedor na iteração n e η é a taxa de aprendizagem do algoritmo, que deve iniciar com um valor η_0 e então decrescer gradualmente com o aumento do tempo.

2.2 Redes neurais auto-organizáveis temporais

A proposta original da rede de Kohonen identifica agrupamentos de dados que estão relacionados entre si de forma espacial. Porém, existem aplicações onde é importante identificar também correlações temporais. No caso do processamento de dados sensoriais para fins de auto-localização, aplicação típica da robótica, o histórico dos dados passados tem relação com os lugares recentemente visitados pelo robô, e fornece uma importante pista para determinação de sua localização atual. Em outras palavras, um determinado padrão na entrada não pode ser interpretado isoladamente. Deve-se levar em conta também os valores que o antecederam. Nesta seção, serão descritas brevemente algumas redes de natureza auto-organizável capazes de extrair correlações temporais. Uma análise mais completa sobre este tema pode ser encontrada em (BASTOS, 2007).

A rede TDSOM, ou Time Delayed Self-Organizing Map (KANGAS, 1990), é uma rede de Kohonen convencional que utiliza memórias de curto prazo na entrada. Ou seja, cada componente do vetor de entrada é substituído por uma sequência que contém a informação temporal, utilizando uma memória de linha de atraso ou traço exponencial. A memória de curto prazo aplicada na entrada converte a informação temporal em espacial, e permite que a rede opere levando em conta o contexto temporal sem que seja necessária nenhuma alteração no seu treinamento ou na forma de representação.

A rede TKM, ou Temporal Kohonen Map (CHAPPEL; TAYLOR, 1993) aplica uma memória da saída da rede. Cada neurônio possui em sua saída um integrador. Desta maneira, o valor da ativação do neurônio em um tempo t é igual ao valor em $t-1$, descontado por uma taxa de decaimento e somado ao valor da função discriminante no tempo t. Tal rede foi empregada com sucesso no reconhecimento de sequências de caracteres. Porém

sua capacidade discriminatória é limitada pela profundidade da memória. Há uma variação da rede TKM onde os integradores são colocados na entrada. Tal rede é chamada de RSOM (VARSTA; HEIKKONEN; MILLáN, 1997), e produz resultados semelhantes à TKM.

O Mapa Auto-Organizável Contextual (CSOM), descrito em (VOEGTLIN, 2000), inclui o aspecto temporal utilizando conexões de realimentação da atividade dos neurônios. Cada neurônio da rede está ligado aos sinais de entrada $\mathbf{x}(t)$ através de um vetor de pesos $\mathbf{w}_i^x$ e também à saída dos neurônios no tempo anterior ($\mathbf{y}(t-1)$), através do vetor de pesos $\mathbf{w}_i^y$. A ativação do i-ésimo neurônio da rede no tempo t é calculada a partir do erro de quantização $E_i(t)$, definido pela equação 2.4:

$$E_i(t) = \alpha||\mathbf{x}(t) - \mathbf{w}_i^x(t)||^2 + \beta||\mathbf{y}(t-1) - \mathbf{w}_i^y(t)||^2 \tag{2.4}$$

onde α e β são duas constantes de esquecimento que devem ter valor entre 0 e 1. A informação de contexto permite que os neurônios se agrupem segundo as características espaciais do padrão atual e também conforme a classificação do padrão anterior. Os pesos são ajustados durante o processo de treinamento de forma análoga à rede SOM convencional.

Nas redes auto-organizáveis discutidas até agora, a classificação do sinal de entrada é representada pelo neurônio vencedor. A rede SARDNET (JAMES; MIIKKULAINEN, 1995) difere-se delas neste ponto: a representação de uma sequência é dada pela configuração de todos os neurônios da rede. Desta forma, aumenta a capacidade de representação de sequências temporais. A cada passo da operação da rede, o neurônio vencedor tem atribuído o valor 1 à sua saída. A saída dos demais neurônios é diminuída através de um fator de decaimento. Para que seqüências temporais repetitivas não gerem ambiguidade na representação, a rede impede que o protótipo vencedor se repita de uma iteração para a seguinte. Isto faz com que a rede possa ter dois protótipos representando o mesmo valor de entrada.

A rede ARAVQ (LINAKER; NIKLASSON, 2000), apesar de não utilizar o conceito de vizinhança existente no mapa de Kohonen e em suas variações, compartilha os conceitos de aprendizado competitivo e pode ser usada para aplicações semelhantes. Outra característica que a distingue é a possibilidade de inclusão de novos neurônios na rede, conforme necessário. A entrada da rede é uma média móvel do sinal de entrada, calculada sobre uma janela de tempo finita τ. A cada iteração, é selecionado o neurônio vencedor: aquele cujo vetor de pesos possui a menor distância euclidiana em relação à média móvel do sinal de entrada. Caso o erro de quantização seja pequeno, o neurônio vencedor tem seus pesos ajustados em direção ao sinal de entrada. Caso contrário, é criado um neurônio novo na rede, cujos pesos são inicializados com o valor da média móvel no instante t.

2.3 Redes neurais auto-organizáveis construtivas

Nesta seção serão analisados alguns exemplos de redes que, além de possuirem a característica de mapa auto-organizável, são capazes de modificar sua estrutura de acordo com a distribuição dos dados de entrada. Tal característica retira do projetista o ônus de determinar previamente a configuração de rede adequada à resolução de um determinado problema. Além disso, permite uma maior flexibilidade, à medida que a mesma rede pode ser aplicada a problemas diferentes sem necessidade de modificação, já que o próprio processo de treinamento encarrega-se de dimensioná-la de forma adequada.

Uma das primeiras redes construtivas foi elaborada por Fritzke (1992), e chama-se

Growing Cell Structures (GCS). Ela consiste em um arranjo bidimensional de neurônios conectados entre si na forma de triângulos. A rede é limitada a um número máximo de simplexos k-dimensionais, onde k é um número inteiro escolhido antes de se iniciar o treinamento. Ao longo do processo de treinamento, cada neurônio guarda a informação do erro acumulado de quantização. Tal informação é útil para determinar onde devem ser incluídas novas unidades. O procedimento de inserção é realizado quando o número de padrões apresentados à rede é múltiplo de um parâmetro λ. A nova unidade é inserida entre o neurônio com o maior erro acumulado e seu vizinho mais distante. Depois dessa primeira unidade, o simplexo é completado de tal forma que a rede resultante estará formada exclusivamente por simplexos k-dimensionais. Entre as variações da rede GCS, pode-se citar a HiGS (BURZEVSKI; MOHAN, 1996) e TreeGCS (HODGE; AUSTIN, 2001), que criam estruturas de submapas hierárquicos.

Uma característica do GCS é a de eventualmente criar arranjos em espaços de elevada dimensão, dificultando a visualização. O modelo IGG - Incremental Grid Growing (BLACKMORE, 1995) foi criado com o intuito de resolver esta questão. A rede é inicializada com 4 neurônios conectados entre si formando um quadrado. Ao longo do processo de treinamento, novas unidades são adicionadas nos limites externos do mapa. Cada nova unidade é conectada ao neurônio de fronteira do qual se derivou. Conexões entre os neurônios são inseridas e removidas conforme a distância euclidiana entre seus vetores de pesos. A estrutura gerada assegura uma fácil visualização das amostras do vetor de entrada.

Similar ao GCS, o modelo Growing Grid (FRITZKE, 1995a), ou simplesmente GG, também cria estruturas de fácil visualização. A principal diferença é que as inserções são feitas sempre na forma de uma nova linha ou coluna de neurônios, mantendo a topologia estritamente retangular. O algoritmo opera em duas fases distintas. Durante a fase de crescimento, novas linhas e colunas vão sendo inseridas até que um critério de parada seja satisfeito. Diferentemente da rede de Kohonen tradicional, a largura da vizinhança é um valor relativamente baixo e constante no tempo. Na fase de convergência, o tamanho da rede não é mais alterado e a taxa de aprendizagem sofre um decaimento exponencial para realizar um ajuste fino na rede.

Na rede GNG - Growing Neural Gas (FRITZKE, 1995b) não apenas o tamanho da rede varia, como a topologia também é determinada a partir dos dados de entrada. O treinamento da rede inicia com apenas dois neurônios conectados entre si. Para cada padrão apresentado à rede, as duas unidades mais próximas são selecionadas, e uma conexão entre elas é criada, caso ainda não exista. A unidade vencedora, juntamente com sua vizinhança, sofre a atualização dos pesos sinápticos. Novas unidades são criadas e removidas durante o processo de treinamento. A inserção de novos neurônios, porém, só ocorre em iterações múltiplas de uma constante predefinida. A rede é capaz de criar estruturas de dados bastante complexas, inclusive com diferentes dimensões e regiões desconexas. Porém, sua convergência é lenta, devido à restrição na inserção de novos neurônios. Além disso, possui dificuldade em se adaptar rapidamente a dados com distribuições que mudam rapidamente (dados não-estacionários).

2.4 A rede GTSOM

A rede GTSOM (BASTOS, 2007) une as características dos mapas auto-organizáveis temporais com as redes construtivas. Trata-se de uma extensão do mapa auto-organizável de Kohonen, nos seguintes pontos:

- A topologia da rede não é fixa, e sim determinada ao longo do treinamento
- Relações temporais entre os dados são levadas em conta, através da utilização de uma memória de curto prazo aplicada na entrada da rede
- A taxa de aprendizado é determinada automaticamente

A seguir serão descritos os pontos mais importantes do funcionamento da rede GTSOM, que são fundamentais para a compreensão deste trabalho.

O vetor que é processado pela rede possui duas partes. A primeira corresponde aos dados de entrada propriamente ditos, ou seja, ao padrão que está sendo apresentado para a rede no instante atual. A segunda é composta por uma memória de curto prazo, a partir de todos os padrões anteriormente apresentados. Desta maneira, o vetor utilizado possui o dobro do número de componentes do vetor de entrada. A memória utilizada é do tipo diferenciadora-integradora (MOSER, 2004). Seu funcionamento é dado pela equação 2.5:

$$\begin{aligned} \overline{\mathbf{x}}(t) &= (1-\mu)\mathbf{x}(t) + \mu\overline{\mathbf{x}}(t-1) \\ \mathbf{y}(t) &= (1-\eta)(\mathbf{x}(t) - \overline{\mathbf{x}}(t-1)) + \eta\mathbf{y}(t-1) \end{aligned} \tag{2.5}$$

Onde μ e η são duas constantes de integração que devem ter valor entre 0 e 1 e $y(t)$ é a saída da memória no instante t. Consideram-se as condições iniciais $\overline{x}(0) = 0$ e $y(0) = 0$.

Tal estrutura, por ser uma memória de curto prazo, poderia ser utilizada na aplicação a que se refere este trabalho para detectar aspectos como a velocidade de deslocamento do robô, que podem ser verificados analisando os dados de entrada mais recentes. Porém, ela não serve para eliminar as ambiguidades sensoriais que aparecem nos ambientes utilizados nos experimentos, como será visto adiante. Por isso, no presente trabalho, a unidade de memória não foi utilizada, adotando-se outra estratégia para a localização.

Cada neurônio possui uma unidade de habituação associada. Esta unidade é utilizada nos processos de inclusão e remoção de neurônios para controlar os nodos que estão sendo utilizados. Ela é uma versão simplificada e discreta do modelo de Habituação de DeLiang Wang e Michael Arbib (WANG, 1998). Matematicamente, a unidade de habituação tem o seu funcionamento regido pela Equação 2.6:

$$\mathbf{w}^h(t) = \mathbf{w}^h(t-1) + \gamma(\alpha(1 - \mathbf{w}^h(t-1)) - \beta\mathbf{w}^h(t-1)\mathbf{S}(t)) \tag{2.6}$$

onde α define a taxa de recuperação da habituação, β controla a velocidade de habituação, γ governa a variação da taxa de habituação e $\mathbf{S}$ é um vetor que representa o estímulo recebido de uma fonte externa no instante t. O valor do estímulo externo aplicado sobre o neurônio vencedor e seus vizinhos imediatos é determinado a partir da distância do protótipo do neurônio para o valor de entrada, conforme a equação 2.7:

$$S_i(t) = exp(-||\mathbf{z}(t) - \mathbf{w}_i(t)||) \tag{2.7}$$

onde i é o índice do neurônio, $\mathbf{z}(t)$ é o vetor formado pela entrada mais o conteúdo da memória, e $\mathbf{w}_i(t)$ é o vetor de pesos do neurônio i.

Os neurônios que não participam da vizinhança imediata do vencedor recebem 0 como estímulo externo. A partir das equações 2.6 e 2.7, nota-se que neurônios cujos protótipos são próximos ao valor de entrada recebem um estímulo próximo de 1, e sua unidade tende ao valor de habituação (0). Os neurônios cujos protótipos estão sempre distantes do valor de entrada tendem ao estado de não habituação (1).

Aqui cabe fazer uma observação com relação ao valor do estímulo externo. Para um correto funcionamento da rede, neurônios com protótipos próximos ao vetor de entrada devem receber estímulo próximo de 1. Porém o conceito de proximidade pode variar dependendo da aplicação. Se a magnitude dos valores da entrada for grande ou o vetor de entrada possuir um grande número de componentes, pode-se desejar que a função seja mais suave. Por isso, no presente trabalho optou-se por uma função um pouco mais genérica para calcular o valor do estímulo externo:

$$S_i(t) = exp(-a||\mathbf{z}(t) - \mathbf{w}_i(t)||) \tag{2.8}$$

onde a é uma constante que controla o decaimento da curva.

A cada iteração, é feita a escolha do neurônio vencedor v, cujo vetor de pesos possui a menor distância euclidiana com relação ao vetor de entrada. Então, é realizado o ajuste dos pesos do neurônio vencedor e de todos os outros neurônios que possuem ligação direta com ele. O ajuste de pesos do neurônio i se dá conforme a equação 2.9:

$$\mathbf{w}_i(t+1) = \begin{cases} \mathbf{w}_i(t) + e_v(t)[\mathbf{z}(t) - \mathbf{w}_i(t)] & \text{se } i = v \\ \mathbf{w}_i(t) + e_n(t)[\mathbf{z}(t) - \mathbf{w}_i(t)] & \text{se } i \text{ é vizinho imediato de } v \\ \mathbf{w}_i(t) & \text{caso contrário} \end{cases} \tag{2.9}$$

A regra de ajuste é semelhante à do mapa de Kohonen convencional. Porém, a taxa de aprendizado não é um parâmetro definido pelo projetista, e sim calculada pelo próprio algoritmo a cada instante de tempo, conforme a equação 2.10, desenvolvida por (BERGLUND; SITTE, 2006).

$$\begin{aligned} p(t) &= max(||\mathbf{z}(t) - \mathbf{w}_v(t)||, p(t-1)) \\ e_v(t) &= \tfrac{||\mathbf{z}(t) - \mathbf{w}_v(t)||}{p(t)} \\ e_n(t) &= \tfrac{e_v(t)}{10} \end{aligned} \tag{2.10}$$

$p(t)$ corresponde à maior distância encontrada até a iteração atual entre o vetor de pesos do neurônio vencedor e o vetor de entrada. $e_v(t)$ é a taxa de aprendizado utilizada para o neurônio vencedor e $e_n(t)$ é a taxa de aprendizado dos vizinhos imediatos do neurônio vencedor. Na primeira iteração, a distância entre o vetor de pesos do vencedor e o vetor de entrada será igual à distância máxima, resultando em $e(v) = 1$. Desta forma, o vetor de pesos do vencedor é substituído pelo vetor de entrada. Conforme a rede vai evoluindo, a tendência é que o protótipo do vencedor fique cada vez mais próximo ao vetor de entrada. Desta forma, o valor da taxa diminui e a rede realiza o ajuste fino dos pesos.

Quando o erro de quantização for alto e o padrão de entrada for relativamente novo para a rede, ocorre a inclusão de um neurônio novo. Matematicamente, a inclusão de um neurônio ocorre quando a seguinte condição é satisfeita:

$$exp(-||\mathbf{z}(t) - \mathbf{w}_v(t)||) < \rho \text{ e } w_v^h(t) < \delta \tag{2.11}$$

onde ρ é o critério de plasticidade e δ é o critério de novidade. O critério de plasticidade define a granularidade da rede (especificidade x generalidade) através de uma medida de erro mínimo aceitável baseada na atividade instantânea do neurônio vencedor. O critério de novidade determina uma medida de novidade baseada no valor da sinapse de habituação do neurônio vencedor. Aqui cabe a mesma observação que foi feita com relação

à equação 2.7. Dependendo da aplicação, pode ser necessário acrescentar um fator de escala na função exponencial.

Já a remoção de um neurônio da rede pode ocorrer em duas situações: quando o neurônio não estiver conectado a nenhum outro, ou quando sua sinapse de habituação estiver em um valor muito próximo a 1, indicando que ele não está sendo utilizado.

A cada padrão de entrada que é processado, a rede identifica os dois neurônios cujos protótipos estão mais próximos do valor de entrada. Caso não haja uma conexão entre eles, esta conexão é criada. Cada conexão possui um contador associado que controla sua idade. A cada passo de tempo, o valor do contador é incrementado. Ele somente é zerado quando os neurônios que ele liga são considerados vencedores. Quando o contador de idade atinge um certo limiar (i_{max}) a conexão é eliminada.

2.5 Considerações do Capítulo

Neste capítulo foi feita uma breve revisão acerca das redes neurais auto-organizáveis, com o objetivo de situar aquela que foi escolhida dentro das alternativas possíveis. A rede GTSOM foi escolhida devido a suas características únicas, como o uso da unidade de habituação para controlar o processo de crescimento e poda, e também porque o próprio objetivo do trabalho consiste em avaliar esta rede.

Destaca-se, porém, que o aspecto temporal desta rede não está sendo utilizado aqui. Como a memória utilizada na rede GTSOM é de curto prazo, ela não é útil para resolver os problemas de ambiguidade sensorial que serão mostrados nos experimentos, e desta forma não contribui para o resultado do aprendizado.

3 MAPEAMENTO E REPRESENTAÇÃO DE AMBIENTES

Mapeamento é o problema de gerar mapas a partir de medidas lidas dos sensores (THRUN, 2000). O robô deve percorrer o ambiente e, enquanto faz isso, criar uma representação interna que facilite o seu deslocamento futuro, permitindo que ele faça uso de informações que não estão ao alcance imediato dos sensores.

Os mapas podem ser classificados em dois tipos: mapas métricos e mapas topológicos. Dentre os mapas métricos, destaca-se o tipo baseado em grade de ocupação, que será analisado a seguir. Também serão estudados os mapas topológicos e algumas abordagens híbridas, que tentam combinar o melhor dos dois tipos.

3.1 Mapas de Grade

Os mapas de grade mantém uma representação métrica do ambiente. Tipicamente, o mapa tem a forma de uma matriz bidimensional, onde cada célula representa uma área do ambiente. A idéia é a mesma de um mapa construído em escala: células adjacentes representam regiões adjacentes no espaço real. O valor armazenado em cada posição da matriz corresponde à probabilidade de encontrar um obstáculo na região correspondente.

A construção do mapa utiliza um método probabilístico incremental. A incerteza na representação vem de duas fontes: o erro de medida dos sensores e o erro de localização do robô. Cada vez que o robô visita um local, as estimativas do mapa são atualizadas com o valor lido dos sensores, levando em conta a estimativa anterior. Desta forma, a incerteza diminui com o passar do tempo. Porém, deve-se ressaltar que isso só é válido com a premissa de um ambiente estático.

A resolução do mapa é definida pelo tamanho de cada região. Mapas com alta resolução permitem maior precisão na representação do ambiente, mas aumentam o custo computacional de elaboração do mapa. Por outro lado, mapas com resolução baixa requerem menos computação, mas aumentam o risco de choque do robô com um obstáculo devido à baixa precisão na representação.

Este é um paradigma que vem sendo utilizado há bastante tempo. (MORAVEC; ELFES, 1985) relata a construção de um mapa de ocupação a partir de leituras de sonar. Foi utilizado um anel com 24 sensores. Cada um deles faz a varredura em um ângulo de 30^{o}, e retorna a distância R em que se encontra o objeto que ocasionou reflexão. A área de varredura do sensor, que corresponde a um cone, é então projetada no plano horizontal, e dividida em duas regiões: a região vazia e a região possivelmente ocupada (ver figura 3.1). A cada uma dessas regiões, é associada uma função de probabilidade. No caso da região possivelmente ocupada, a função determina a probabilidade de que exista um objeto no local. A probabilidade é maior nos pontos cuja distância em relação à posição do sensor é mais próxima de R. Já na região vazia, outra função determina a probabilidade de que

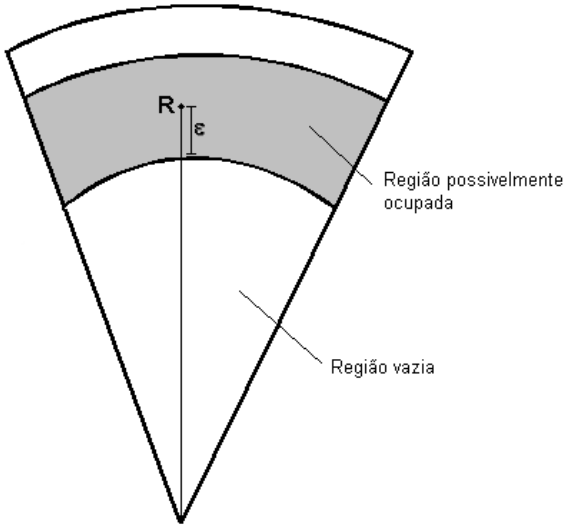

Figura 3.1: Varredura do sonar (MORAVEC; ELFES, 1985)

não exista um objeto. Esta probabilidade é maior nos pontos mais próximos ao sensor.

Cada célula no mapa possui dois valores associados, correspondentes à crença de que a região esteja vazia ou ocupada. A cada leitura do sensor, o valor calculado da probabilidade para cada um dos pontos é combinado com o valor anterior que estava armazenado no mapa nos pontos correspondentes. Se duas leituras fornecem a mesma informação para um determinado ponto, o resultado é o aumento da probabilidade de que a região esteja ocupada/vazia. Já se a informação que está sendo incorporada difere da que estava armazenada no mapa, o resultado é o aumento da incerteza a respeito da região. É importante notar que, para fazer a correspondência dos pontos obtidos na leitura do sensor com os pontos armazenados no mapa é necessário que o robô saiba a sua localização. Isto caracteriza o problema da auto-localização.

Os mapas de grade também podem ser construídos utilizando informações de outros tipos de sensores, como infra-vermelho, laser, ou câmeras de vídeo. No caso da utilização de câmeras, é necessário realizar o processamento da imagem para detecção de obstáculos tornando o processo mais custoso.

Segundo Browning (2000), a representação por mapas de grade representa uma abordagem concisa e fácil para o mapeamento. Sistemas com sólido embasamento probabilístico são capazes de operar com sucesso mesmo em ambientes complexos e dinâmicos. Além disso, a natureza quantitativa da representação permite que sistemas de planejamento de trajetórias construam rotas ótimas para atingir o destino. Por outro lado, possui a desvantagem do custo computacional elevado. A grande quantidade de memória necessária ao armazenamento dos mapas resulta em baixa escalabilidade para ambientes grandes.

3.2 Mapas Topológicos

Os mapas topológicos são também chamados de mapas qualitativos. Eles modelam o mundo como um conjunto de lugares distintos, com conexões entre eles. Existem vários

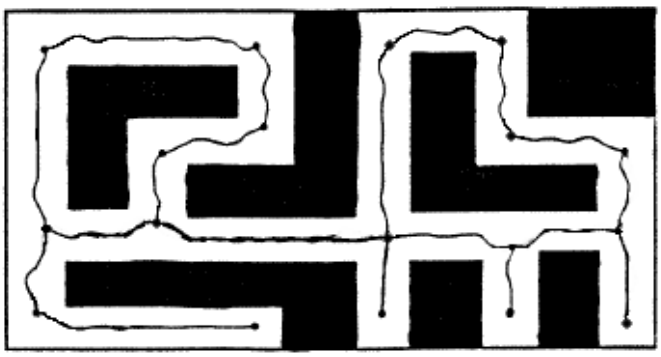

Figura 3.2: Mapa topológico (KUIPERS; BYUN, 1988)

modelos deste tipo, que diferem na maneira de definir o que é um lugar distinto, e na forma de armazenamento do mapa.

Um método de representação por mapa topológico é descrito em (KUIPERS; BYUN, 1988). Nele, os vértices do grafo são definidos como os máximos locais de uma medida construída a partir dos dados sensoriais, como por exemplo a diferença das distâncias entre objetos próximos. Para identificar estes locais, o robô utiliza um método de subida de gradiente durante a fase de exploração. Os arcos do grafo consistem em procedimentos que descrevem como se deslocar de um ponto a outro, juntamente com informações métricas, como a distância do percurso. Estas informações são atualizadas constantemente enquanto o robô desloca-se pelo ambiente. Um exemplo de mapa topológico formado a partir deste procedimento é mostrado na figura 3.2.

Outra maneira de representar os mapas topológicos é utilizando cadeias de Markov (SHATKAY; KAELBLING, 1997). Nesta abordagem, os lugares no ambiente correspondem a estados no modelo de Markov. As conexões entre os lugares são modeladas como uma série de estados próximos, correspondendo ao caminho que deve ser percorrido. O movimento do robô é visto como um processo de Markov parcialmente observável onde as ações correspondem a transições probabilísticas de estados. Por ser um modelo probabilístico, este método consegue lidar com as incertezas nos atuadores do robô.

Existem ainda trabalhos que descrevem a implementação de mapas topológicos utilizando redes neurais auto-organizáveis do tipo SOM, como por exemplo (OWEN; NEHMZOW, 1998). A rede recebe como entrada os valores dos sensores e uma bússola externa para orientação. Ao longo da fase de treinamento, a rede forma agrupamentos de dados, de forma que cada nodo na rede corresponde a um lugar no ambiente. Além disso, nodos que estão conectados no mapa representam regiões próximas no ambiente.

Os mapas topológicos, em comparação com os modelos de grade de ocupação, não sofrem da mesma limitação quanto ao tamanho do ambiente. Como a representação se dá por características, o aumento no tamanho do ambiente não implica em grande aumento na memória necessária para representação. Porém, é bom ressaltar que isso pode ser uma desvantagem. No caso de duas salas ligadas por um corredor, por exemplo, o mapa topológico representaria cada sala como um único ponto, o que pode inviabilizar a execução de tarefas envolvendo vários locais dentro da mesma sala. A tabela 3.1 resume as vantagens e desvantagens dos dois tipos de representação.

	Mapas de Grade		Mapas Topológicos
+	fáceis de construir, representar e atualizar	+	permitem planejamento eficiente e ocupam pouca memória
+	eliminam ambiguidade no reconhecimento de locais já visitados	+	não necessitam que a posição do robô seja conhecida com precisão
+	permitem cálculo do caminho mais curto	+	são convenientes para uso com planejadores simbólicos
-	planejamento computacionalmente custoso	-	difíceis de construir e manter em ambientes grandes se a informação sensorial é ambígua
-	requerem determinação precisa da posição do robô	-	reconhecimento de lugares já visitados é difícil, e sensível ao ponto de vista
-	integração com planejadores simbólicos é difícil	-	podem gerar caminhos subótimos

Tabela 3.1: Comparação dos mapas de grade e topológicos (THRUN, 1998)

3.3 Modelos Híbridos

Muitos pesquisadores tentaram juntar as vantagens da abordagem quantitativa (mapas de grade) com a abordagem topológica construindo sistemas híbridos. Desta maneira, objetiva-se unir a precisão dos mapas de grade com a eficiência dos mapas topológicos. Há muitos trabalhos nesta linha, com abordagens diferentes. Nesta seção serão descritos dois deles.

Thrun (1998) relata a construção de um modelo com este objetivo. O robô primeiro constrói o mapa de ocupação através da exploração do ambiente, para em seguida gerar um mapa topológico a partir da grade de ocupação. O procedimento que gera o mapa topológico consiste em decompor o mapa em uma série de regiões separadas por passagens estreitas, como portas. Considerando as regiões como nodos de um grafo e as portas como as conexões, obtém-se o mapa topológico da região explorada. O autor relata que a construção do mapa topológico desta forma evita dois problemas que são comuns nos sistemas que constroem o mapa topológico diretamente a partir dos dados sensoriais: a ambigüidade (lugares diferentes com aparência semelhante) e o estabelecimento de correspondência (diferentes visões do mesmo lugar).

Outra maneira de combinar as abordagens quantitativa e qualitativa é construir vários mapas de grade, correspondentes a regiões diferentes e conectá-los através de um modelo topológico. Esta foi a idéia adotada em (CHONG; KLEEMAN, 1999). No modelo desenvolvido, novos mapas locais são criados dinamicamente sempre que a covariância da posição do robô excede um limiar. A posição relativa de cada mapa local é armazenada em um grafo topológico. Este método é capaz de mapear grandes áreas de forma mais eficiente do que se fosse usado um único mapa.

3.4 Considerações do Capítulo

Neste capítulo foram mostradas as principais abordagens para construção de mapas. Optou-se, neste trabalho, pela utilização de um mapa de grade para representação do ambiente. A utilização do mapa em conjunto com os agrupamentos sensoriais faz-se necessário dentro do modelo proposto para solucionar os casos onde a percepção sensorial é idêntica em lugares diferentes do ambiente. E o mapa de grade é adequado para integração com o modelo proposto, pois sua representação em forma de coordenadas pode facilmente ser utilizada para criar agrupamentos, correspondentes a regiões no espaço, segmentando o espaço de estados. Esse assunto será discutido em detalhes quando da apresentação dos resultados experimentais.

4 SISTEMAS DE NAVEGAÇÃO COM APRENDIZADO POR REFORÇO

Esta seção descreve alguns trabalhos que se assemelham ao modelo que está sendo proposto, por fazerem o uso de técnicas de aprendizado por reforço para resolução de tarefas de navegação de robôs. Inicialmente, será feito um breve apanhado dos fundamentos teóricos e dos principais algoritmos associados a esta forma de aprendizado. Em seguida, serão expostas as aplicações relacionadas, salientando os pontos de maior interesse para o presente estudo.

4.1 Fundamentos

Aprendizado por reforço é o problema enfrentado por um agente que deve aprender a se comportar por tentativa e erro, através da interação com o ambiente (KAELBLING; LITTMAN; MOORE, 1996). Diferentemente do que acontece no aprendizado supervisionado, não existe um professor capaz de determinar qual a melhor ação a ser tomada em cada situação. O agente deve utilizar sua conexão sensório-motora com o ambiente para inferir relações de causa e efeito, sobre as consequências de suas ações e sobre o que ele deve fazer para atingir seus objetivos. O aprendizado por reforço é definido não como um método de aprendizado, mas sim como um problema a ser resolvido. Todos os métodos adequados à solução deste problema são considerados métodos de aprendizado por reforço.

Um dos pontos que faz com que o problema do aprendizado por reforço seja de difícil solução é a possibilidade de que a resposta do ambiente seja consequência não apenas da última ação executada, mas também de ações tomadas há mais tempo. Desta forma, as recompensas recebidas pelo agente podem ser defasadas de vários instantes de tempo em relação às ações que as ocasionaram. Estas duas propriedades - busca por tentativa e erro e recompensa defasada - são as características marcantes do aprendizado por reforço.

Além do agente e do ambiente, um sistema de aprendizado por reforço contém 4 elementos (SUTTON; BARTO, 1998): uma política, uma função de recompensa, uma função de valor e, opcionalmente, um modelo do ambiente.

A política define a maneira com que o agente se comporta. Pode ser vista como um mapeamento do estado do ambiente, conforme percebido pelo agente, para ações a serem tomadas em cada estado. Pode ser uma simples tabela, ou pode envolver um planejamento mais elaborado. Também é possível que seja estocástica. A política é o centro do agente no Aprendizado por Reforço, pois ela sozinha é suficiente para definir o comportamento.

A função de recompensa define o objetivo do agente. A cada estado (ou par estado-ação), esta função associa um valor que simboliza o quanto é bom ou ruim para o agente

estar naquele estado. Assim como uma pessoa utiliza estímulos como prazer e dor para modificar seu comportamento, o agente deve desenvolver uma política que maximize a recompensa total recebida.

Enquanto a função de recompensa indica o que é bom ou ruim no tempo atual (imediato), a função de valor indica quais os melhores estados levando em consideração os estados que podem vir em seguida e os valores das recompensas futuras. Ou seja, ele depende da soma das recompensas que o agente poderá receber no futuro, a partir daquele estado. Fazendo novamente a analogia com os humanos, o valor do estado pode ser visto como a satisfação a respeito do atual estado do ambiente após um julgamento elaborado considerando o que pode vir no futuro.

O último elemento é o modelo do ambiente. Ele compreende as probabilidades de transição de estado. Dado um estado atual e uma ação executada, o modelo pode prever qual deve ser o próximo estado do ambiente. Quanto ao conhecimento do agente sobre o modelo do ambiente, existem 3 possibilidades:

- O agente possui um modelo completo do funcionamento do ambiente, fornecido a priori pelo projetista do sistema;
- O agente aprende um modelo através da interação com o ambiente;
- O agente opera e desenvolve sua política sem a necessidade de um modelo do ambiente.

Para cada uma destas situações, existem métodos de aprendizado adeqüados.

4.1.1 O problema do Aprendizado por Reforço

Nesta seção será feita a formalização do problema do aprendizado por reforço. Será tratada aqui a versão de tempo discreto, embora muitas das idéias possam ser estendidas para o caso contínuo (DOYA et al., 2000).

O agente interage com o ambiente em uma sequência de passos de tempo discretos $t = 0, 1, 2, ...$ A cada passo de tempo, o agente recebe uma representação do estado do ambiente $s_t \in S$, onde S é o conjunto de todos os estados possíveis. Ele então seleciona uma ação $a_t \in A(s_t)$, onde $A(s_t)$ é o conjunto de ações disponíveis no estado s_t. No tempo seguinte, o agente recebe, em parte como consequência de sua ação, uma recompensa numérica $r_{t+1} \in \Re$ e se encontra em um novo estado s_{t+1}. Esta situação pode ser vista na figura 4.1.

O agente implementa um mapeamento de estados para probabilidades de seleção de cada uma das ações possíveis. Este mapeamento é a política π_t, onde $\pi_t(s, a)$ é a probabilidade de que $a_t = a$ quando $s_t = s$. Pela aprendizagem, o agente muda esta política com o objetivo de maximizar a recompensa.

Este *framework* é flexível e pode ser adaptado de acordo com as particularidades do problema. Por exemplo, os passos de tempo não precisam necessariamente referir-se ao tempo real, mas podem indicar as etapas do aprendizado, ativado por algum evento externo. As ações podem ser comandos de baixo nível, como uma voltagem aplicada a um motor, ou podem ser decisões como ir para frente ou virar à esquerda. Da mesma forma, os estados podem ser os próprios valores lidos dos sensores, ou podem ser uma representação de mais alto nível. O estado também pode levar em conta a memória de sensações passadas.

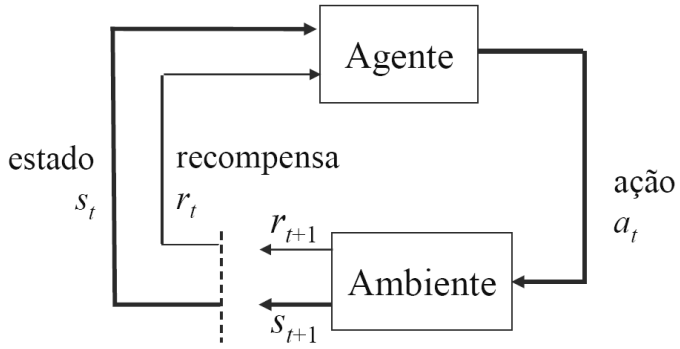

Figura 4.1: A interação agente-ambiente (SUTTON; BARTO, 1998)

4.1.1.1 Objetivos e Recompensas

No aprendizado por reforço, o objetivo é formalizado pelo sinal de reforço, passado pelo ambiente para o agente. O reforço pode ser, por exemplo, +1 quando a ação do agente ocasiona sucesso e -1 caso contrário. Se a tarefa a ser resolvida é escapar de um labirinto, o reforço pode ser 0 enquanto o agente não conseguiu escapar, e +1 quando encontra a saída. Uma prática comum é atribuir um reforço de -1 a cada passo executado - isso encoraja o agente a encontrar o menor caminho.

Um ponto importante é que a função de recompensa deve ser definida pelo projetista de modo a indicar *o que* o agente deve fazer, e não *como*. Caso deseje-se utilizar algum conhecimento prévio sobre como a tarefa deve ser executada, a melhor maneira de fazer isso é na inicialização da política de ações ou da função de valor.

Em geral, o agente é projetado para maximizar o retorno esperado R_t, definido como uma função específica da sequência de reforços $r_{t+1}, r_{t+2}, ..., r_T$ recebidos pelo agente ao longo do tempo. Se a tarefa que está sendo executada possui um estado final, então a interação entre agente e ambiente pode ser dividida em subsequências, chamadas de *episódios*. Um exemplo deste tipo de tarefa é achar a saída em um labirinto. O episódio termina quando o agente encontra a saída ou quando uma condição de parada é atingida. Neste caso, pode fazer sentido definir o retorno esperado como a soma dos reforços:

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + ... + r_T \tag{4.1}$$

Por outro lado, existem tarefas contínuas, que não podem ser divididas em episódios. Neste caso, a definição de retorno esperado como soma simples dos reforços não pode ser aplicada, pois o horizonte de tempo é infinito. Neste caso, utiliza-se o retorno descontado

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{4.2}$$

onde $0 \leq \gamma \leq 1$ é a taxa de desconto. Quando $\gamma = 0$, o agente é dito míope, pois ele só se preocupa em maximizar os reforços imediatos. Conforme γ aumenta, cresce a importância relativa dos reforços futuros.

Por ser mais fácil de manipular matematicamente, a função de soma descontada é normalmente utilizada para definir o retorno esperado tanto em tarefas episódicas como em tarefas contínuas.

4.1.1.2 Processo de Decisão de Markov

Nos sistemas de aprendizado por reforço, as decisões do agente são tomadas em função de um sinal de estado recebido do ambiente. Os algoritmos de aprendizado não abordam o problema da construção desta representação do estado, partindo do pressuposto de que tal informação está disponível. A determinação do estado em que se encontra o ambiente, a partir das informações que estão disponíveis para o agente, é fundamental para o correto aprendizado e é o ponto central deste estudo.

O sinal de estado deve incluir informações imediatas, como leituras de sensores, e possivelmente alguma informação adicional. Estas informações adicionais podem ser representações de alto nível obtidas através de processamento das leituras sensoriais, ou extraídas a partir do histórico de sensações passadas. Por exemplo ao fazer uma observação de um objeto em um tempo t, é possível saber sua posição. Porém, para saber a velocidade, é preciso combinar a informação as observações de pelo menos dois instantes de tempo.

Idealmente, o estado atual deve conter toda a informação relevante sobre as sensações passadas. Isto normalmente requer mais do que a informação disponível no tempo atual, mas certamente não mais do que o histórico completo. Quando o sinal de estado satisfaz a esta condição, diz-se que ele possui a *propriedade de Markov*.

Quando o ambiente satisfaz à propriedade de Markov, é possível fazer a previsão do próximo estado e da recompensa que será recebida a partir do estado atual e da ação executada. Não importa quais os estados que ocorreram anteriormente, nem as ações que foram executadas até então, pois o estado atual contém toda a informação necessária. Por isso, ao construir uma política de ações, podemos utilizar o somente o estado como parâmetro.

Não se deve pensar, por outro lado, que a conformidade com a propriedade de Markov é suficiente para considerar uma representação de estado como adequada à resolução do problema. A presença de informações não relevantes ou de informações com um nível de detalhe exagerado pode causar uma explosão combinatória no número de estados do sistema, inviabilizando o aprendizado. Tomando como exemplo um robô que deve achar a saída de um labirinto, poderíamos utilizar como estado sua posição (x, y), orientação θ e velocidade instantânea v. Desta forma, o estado seria formado pela tupla (x, y, θ, v). Apesar de ser suficiente para permitir a escolha da melhor ação, tal representação pode gerar estados diferentes para situações que, para fins de seleção da ação e previsão do próximo estado, são equivalentes. Por outro lado, uma representação utilizando conceitos de alto nível, como *início do corredor*, *final do corredor* e *intersecção* poderia resolver o mesmo problema de forma mais eficiente.

Uma tarefa de aprendizado por reforço que satisfaz à propriedade de Markov é chamada de Processo de Decisão de Markov (PDM). No caso em que os espaços de estados e ações são finitos, diz-se que é um Processo de Decisão de Markov Finito. Um PDM finito é definido pelos seus conjuntos de estados e ações, e por uma dinâmica do ambiente que leva em conta um único passo de tempo. Dados um estado s e uma ação a, a probabilidade de transição para cada um dos próximos estados possíveis s' é dada por:

$$P^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}, \text{com} \sum_{s'} P^a_{ss'} = 1 \; \forall s \tag{4.3}$$

As transições de estado ocorrem de forma probabilística. A execução da mesma ação em um mesmo estado pode gerar resultados diferentes.

De forma análoga, dado um estado corrente s e uma ação a, juntamente com um próximo estado s', o valor esperado do próximo reforço é:

$$R^a_{ss'} = E\{r_{t+1}|s_t = s, a_t = a, s_t = s'\} \tag{4.4}$$

4.1.1.3 Funções de Valor

Os algoritmos de aprendizado por reforço utilizam funções de valor - que estimam o quão bom é para o agente estar em um estado, ou executar uma certa ação no estado. Quanto maior o valor esperado de recompensas futuras que serão recebidas, maior é o valor do estado.

Como as recompensas que o agente receberá dependem diretamente das ações que ele escolher, a função de valor do estado é definida com respeito a uma determinada política. A política de ações é definida formalmente como uma função $\pi(s, a)$, que mapeia um par estado-ação (s, a) para a probabilidade de que o agente escolha a ação a quando estiver no estado s.

Já o valor do estado para uma determinada política é o retorno esperado quando, a partir do estado s, o agente segue a política π:

$$V^\pi(s) = E_\pi\{R_t|s_t = s\} = E_\pi\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\Big|s_t = s\right\} \tag{4.5}$$

Analogamente, a função de valor da ação para a política π é definida como o retorno esperado quando o agente, iniciando no estado s, executa a ação a e em seguida segue a política π:

$$Q^\pi(s, a) = E_\pi\{R_t|s_t = s, a_t = a\} = E_\pi\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\Big|s_t = s, a_t = a\right\} \tag{4.6}$$

As funções de valor V^π e Q^π podem ser estimadas a partir da experiência. Se o agente segue a política π e mantém uma média, para cada estado visitado, das recompensas que recebeu a partir dele, então estas médias convergirão para o valor do estado $V^\pi(s)$ à medida que o número de visitas ao estado se aproxima do infinito. Se forem armazenadas médias separadas para cada uma das ações tomadas no estado, então o resultado é o valor da ação $Q^\pi(s)$. Uma estimação de valores como esta, envolvendo médias sobre amostras aleatórias é chamada de método de Monte Carlo.

Uma propriedade fundamental das funções de valor bastante explorada nos métodos de aprendizado por reforço é que elas obedecem a algumas relações recursivas. Dada uma política π e um estado s, vale a seguinte relação entre o valor do estado s e os valores de seus possíveis sucessores:

$$V^\pi(s) = \sum_a \pi(s, a)\sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V^\pi(s')] \tag{4.7}$$

A equação 4.7 é chamada de *Equação de Bellman*. Ela faz a média de todas as possibilidades para o próximo estado, ponderando os valores pela probabilidade de ocorrência.

O objetivo dos algoritmos para solucionar o problema da aprendizagem por reforço é encontrar uma política ótima para o agente. A política ótima π^* é aquela que maximiza a função de valor. A função de valor de estado para a política ótima é a função ótima de valor de estado, denotada como $V^*(s)$, e definida por:

$$V^*(S) = \max_{\pi} V^{\pi}(s) \tag{4.8}$$

4.1.2 Programação Dinâmica

Quando o agente possui de antemão um modelo completo do ambiente (isto é: os conjuntos de estados e ações, a função de transição de estados e a função de recompensa), é possível determinar a política ideal de ações a serem tomadas. O termo *programação dinâmica* refere-se a uma coleção de algoritmos desenvolvidos com esta finalidade. Serão descritos a seguir dois deles: iteração de valor e iteração de política.

Primeiramente, deve-se considerar o problema de calcular a função de valor de estado V^{π} para uma determinada política. Esta tarefa é chamada de *avaliação de política*, ou de *problema da predição*. Ele pode ser resolvido algebricamente, através da solução das equações de Bellman ou iterativamente.

Solucionar as equações de Bellman equivale a resolver um sistema de $|S|$ equações lineares com $|S|$ incógnitas, sendo $|S|$ o número de estados possíveis do ambiente. Embora não seja um processo complexo, na maior parte das vezes é mais interessante utilizar a versão iterativa da avaliação de política.

Para avaliar a política iterativamente, parte-se de uma aproximação inicial arbitrária, V_0. A partir dela, são feitas aproximações sucessivas V_1, V_2, ... V_k. Cada aproximação sucessiva é obtida a partir da estimativa anterior utilizando a equação de Bellman como regra de atualização:

$$\begin{aligned} V_{k+1}(s) &= E_{\pi}\left\{r_{t+1} + \gamma V_k(s_{t+1}) | s_t = s\right\} \\ &= \sum_a \pi(s,a) \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V_k(s')] \end{aligned} \tag{4.9}$$

A cada iteração, são percorridos todos os estados. Para cada um deles, a nova aproximação é calculada utilizando as estimativas atuais dos valores de seus possíveis sucessores ($V_k(s')$), conforme sua probabilidade de ocorrência, descontadas pelo fator γ e somadas aos valores das recompensas associadas às transições de estado. Pode-se provar que, quando $\gamma < 1$, ou quando é garantido que sempre será atingido um estado terminal, o algoritmo de avaliação iterativa de política converge para V_{π}.

A *avaliação de política* pode ser utilizada para melhorar uma política existente. Suponha que a função de valor V^{π} tenha sido calculada para uma política arbitrária. Para um dado estado, podemos questionar se não seria melhor escolher uma ação $a \neq \pi(s)$. Sabemos que, seguindo a política $\pi(s)$, o retorno esperado é $V^{\pi}(s)$. Por outro lado, se selecionarmos inicialmente a ação a para depois seguir a política π, o retorno esperado é dado pela função de valor da ação $Q^{\pi}(s,a)$, conforme a equação 4.6. Se $Q^{\pi}(s,a)$ for maior do que $V^{\pi}(s)$, então pode-se dizer que é melhor selecionar a quando estivermos no estado s e, a partir daí, seguir a política π, do que seria seguir π o tempo todo. Este resultado é garantido pelo *teorema da melhoria de política*, cuja prova pode ser encontrada em (SUTTON; BARTO, 1998).

Uma extensão natural da estratégia de melhoria de uma política é considerar as mudanças para todos os estados, selecionando em cada um deles a melhor ação segundo $Q^{\pi}(s,a)$. Neste caso, a política melhorada será a *política gulosa* π', dada por:

$$\begin{aligned} \pi'(s) &= \arg\max_a Q^{\pi}(s,a) \\ &= \arg\max_a E_{\pi}\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s, a_t = a\} \\ &= \arg\max_a \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V^{\pi}(s')] \end{aligned} \tag{4.10}$$

A política gulosa escolhe a ação que é a melhor a curto prazo (um passo de tempo no futuro) de acordo com V^{π}. O teorema da melhoria de política garante que a política π' construída desta forma é tão boa quanto ou melhor do que a política π.

Uma vez que a política π tenha sido melhorada utilizando V^{π} para gerar uma nova política π', podemos calcular $V^{\pi'}$ e utilizar o resultado para gerar uma política ainda melhor. Repetindo-se esses passos de avaliação e melhoria da política, cria-se uma sequência de políticas $\pi_0, \pi_1, ...$ e funções de valor correspondentes $V^{\pi_0}, V^{\pi_1}, ...$ Como cada política da sequência é melhor do que a anterior e um PDM finito possui um número finito de políticas, garante-se, nestas condições, a convergência da sequência para a política ótima. Este processo é chamado de *Iteração de Política*, e está ilustrado no algoritmo 1.

```
// 1. Inicialização
Escolher V(s) e π(s) arbitrariamente ∀s ∈ S;
// 2. Avaliação de Política
repita
    Δ ← 0;
    para cada s ∈ S faça
        v ← V(s);
        V(s) ← Σ_s' P^π(s)_ss' [R^π(s)_ss' + γV(s')];
        Δ ← max(Δ, |v − V(s)|);
    fim
até Δ < θ ;
// 3. Melhoria de Política
politica_estavel ← verdadeiro;
para cada s ∈ S faça
    b ← π(s);
    π(s) ← arg max_a Σ_s' P^a_ss' [R^a_ss' + γV^π(s')];
    if b ≠ π(s) then
        politica_estavel ← falso;
    end
fim
if politica_estavel then
    parar;
else
    Ir para o passo 2 (Avaliação de Política)
end
```

Algoritmo 1: Iteração de Política

A constante θ que aparece no algoritmo 1 é utilizada para controlar o final do processo de avaliação da política, e deve assumir um valor pequeno e positivo.

Uma desvantagem do algoritmo de Iteração de Política é que cada uma de suas iterações envolve a avaliação da política, que por sua vez pode demandar várias varreduras no conjunto de estados. Se a avaliação da política for feita iterativamente, a convergência ocorre somente no limite. Porém, não é necessário esperar a convergência para melhorar a política. No algoritmo de *Iteração de Valor*, a avaliação da política é truncada após um passo apenas. Ele pode ser visto como uma operação de regressão simples, que combina os passos de melhoria de política e avaliação truncada, conforme a equação 4.11

$$\begin{aligned} V_{k+1}(s) &= \max_a E\{r_{t+1} + \gamma V_k(s_{t+1}) | s_t = s, a_t = a\} \\ &= \max_a \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V_k(s')\right] \end{aligned} \tag{4.11}$$

O algoritmo 2 descreve o método de solução por Iteração de Valor.

Inicializar $V(s)$ arbitrariamente;
repita
  $\Delta \leftarrow 0$;
  para cada $s \in S$ **faça**
    $v \leftarrow V(s)$;
    $V(s) \leftarrow \max_a \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V_k(s')\right]$;
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$;
  fim
até $\Delta < \theta$;
// a saída é uma política π tal que
$\pi(s) = \arg\max \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V_k(s')\right]$;

Algoritmo 2: Iteração de valor

4.1.3 Aprendizado por Diferença Temporal

Os métodos de aprendizado por diferença temporal permitem a construção gradual de uma estimativa da política ótima utilizando apenas a experiência adquirida, sem necessidade de conhecimento do modelo do ambiente. A cada passo de tempo, os valores de recompensa observados são utilizados para atualizar a estimativa V da função de valor V^π do estado que foi visitado. Para o problema de controle, utiliza-se alguma variação da iteração de política gulosa.

O método TD mais simples, TD(0), funciona conforme ilustrado no algoritmo 3.

Inicializar $V(s)$ arbitrariamente e π conforme a política a ser avaliada;
para cada *episódio* **faça**
  Inicializar s;
  para cada *passo do episódio* **faça**
    $a \leftarrow \pi(s)$;
    executar ação a, observar recompensa r e próximo estado s';
    $V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)]$;
    $s \leftarrow s'$
  fim
fim

Algoritmo 3: TD(0)

Outro método TD bastante popular é o Q-Learning. Ele constrói uma estimativa da função de valor da ação $Q(s, a)$, que aproxima diretamente a função ótima $Q^*(s, a)$, independentemente da política que está sendo adotada. O funcionamento do método é mostrado no algoritmo 4.

4.2 Aplicações

Esta seção descreve aplicações que fazem uso de aprendizado por reforço para solucionar problemas de navegação robótica. Dada uma tarefa, o robô deve escolher uma

Inicializar $Q(s, a)$ arbitrariamente;
para cada *episódio* **faça**
  Inicializar s;
  para cada *passo do episódio* **faça**
    Escolher a de s utilizando política derivada de Q (ex: ϵ-gulosa);
    executar ação a, observar recompensa r e próximo estado s';
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma\ max_{a'} Q(s', a') - Q(s, a)]$;
    $s \leftarrow s'$;
    até que s seja terminal;
  fim
fim

Algoritmo 4: Q-Learning

sequência de ações que o permitam deslocar-se até o objetivo de maneira eficiente, evitando colisões. Este é um problema difícil de ser resolvido através dos métodos tradicionais de aprendizado por reforço. O espaço de estados é contínuo e multidimensional. Além disso, o ambiente é parcialmente observável. O que está disponível como entrada através dos sensores do robô não é o estado em si, mas uma observação dele, que pode ser ruidosa e ambígua. Outro ponto que torna difícil o aprendizado é que a recompensa é defasada. Pode ser necessária uma longa sequência de ações até que o robô receba o reforço correspondente. A seguir, serão listados alguns trabalhos nesta linha de estudo.

Em (LI; PANG, 2006) é apresentada uma abordagem para este problema. O espaço de entrada é discretizado por uma rede neural competitiva. O nodo vencedor da rede é utilizado para identificar o estado atual do agente. Desta forma, o algoritmo Q-Learning pode ser aplicado para aprendizado da função de valor das ações.

A rede neural utilizada é baseada na arquitetura da rede de Kohonen. Foram definidas regras que permitem a criação de novos nodos na rede, conforme necessidade e também a junção de dois nodos em um só. Um neurônio é incluído na rede quando duas condições forem satisfeitas:

- A distância euclidiana entre a entrada e o protótipo mais próximo dela deve ser maior que um limiar Δ.
- O erro TD deve ser maior do que um parâmetro δ_{max}

A rede inicia com apenas um nodo e vai crescendo à medida que o agente explora o ambiente. Eventualmente, podem ser criados nodos supérfluos. Por isso, existe uma regra que define quando dois nodos devem ser fundidos. Dois nodos i e j são transformados em um único nó, quando:

- j é o nó mais próximo de i
- a diferença quadrática média entre os valores de ação dos estados i e j é menor do que um parâmetro ρ.

Para aprendizado da função de valor de ação, foi utilizado Q-Learning com traço de eligibilidade. Os autores descrevem um experimento que foi realizado por simulação, em um ambiente de 10 unidades de largura e 10 de altura. Neste ambiente, foi colocado um robô com 8 sensores tipo sonar, com a tarefa de sair de um local de partida, localizado em um canto do ambiente, desviar de obstáculos no caminho e chegar ao destino, no canto

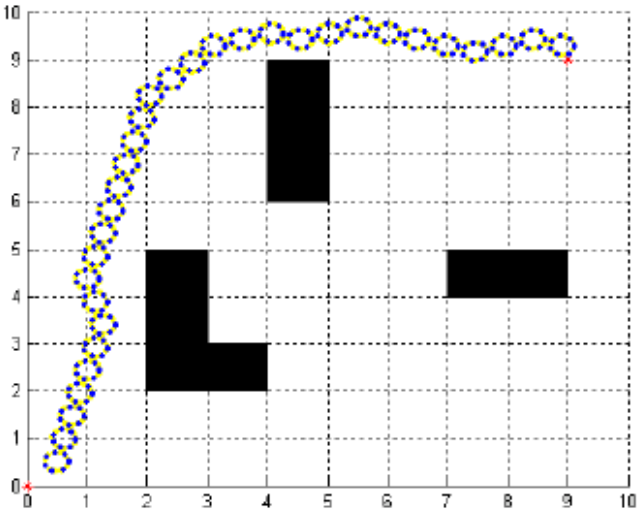

Figura 4.2: Trajetória do Robô (LI; PANG, 2006)

oposto. O reforço recebido é de 1 quando atinge o objetivo, 0.1 por se aproximar dele, -0.1 ao se afastar do objetivo e -1 quando colide com um obstáculo. Com uma configuração inicial favorável, o robô demora cerca de 60 épocas para aprender a tarefa. A trajetória executada pelo robô após o aprendizado pode ser vista na figura 4.2.

O reforço recebido pelo agente quando toma uma ação que se aproxima do objetivo facilita bastante o aprendizado, já que não é necessário chegar até o alvo para que ele descubra que a ação tomada foi a correta. Porém, isso é pouco realista. Em tarefas de exploração do ambiente o robô não sabe se a ação foi correta até que atinja o objetivo, a não ser que esteja sendo treinado de forma supervisionada. Além disso, fica clara a maneira como é resolvida a questão da ambiguidade sensorial. O autor afirma que a entrada da rede é composta pelas 8 leituras dos sonares. Assume que o robô conhece sua posição e orientação ao longo dos episódios, mas não explica de que forma esta informação é utilizada.

5 ADAPTAÇÕES NA REDE GTSOM PARA CATEGORIZAÇÃO DE DADOS SENSORIAIS

Este capítulo relata uma série de estudos realizados com o intuito de avaliar a adequação da rede GTSOM para utilização em aprendizado por reforço, no domínio da robótica. Considera-se um robô dotado de um conjunto de sensores de distância, explorando um ambiente fechado. As leituras dos sensores são ligadas na rede GTSOM. Espera-se que, após um período de adaptação, a rede consiga criar um conjunto de protótipos que representem adequadamente o espaço de entrada. Além disso, é desejável que esta representação seja estável, ou seja: após o nodo ter sido treinado, não deve ocorrer alteração significativa em seu vetor de pesos.

Em (BASTOS, 2007) já foram relatados experimentos com trajetórias, porém há duas diferenças importantes com relação aos que estão descritos aqui:

- Bastos utilizou como entrada da rede a posição (x, y). Aqui estão sendo utilizados como entrada sensores de distância. Seu valor depende não só da posição do agente, mas também da direção angular em que ele está posicionado.

- Neste trabalho, deseja-se ressaltar a diferença entre o equilíbrio dinâmico – onde o número de neurônios e a configuração da rede permanecem fundamentalmente os mesmos, porém os valores individuais dos pesos mudam – e o equilíbrio estático – onde após a convergência, os valores dos pesos não sofrem alterações significativas.

Em todos os experimentos relatados abaixo, o robô foi equipado com um conjunto de 4 sensores de distância: D_0, posicionado à sua frente, D_1 à esquerda, D_2 atrás e D_3 à direita. Assume-se que a não existência de ruído nas medições dos sensores. Ao fazer referência ao vetor composto pela leitura dos 4 sensores, será usada a notação $\mathbf{D}$. O resultado esperado da rede é a criação de nodos que possam ser interpretados como estados de uma cadeia de Markov. Por isso é importante a estabilidade de representação.

5.1 Experimento 1

Neste primeiro experimento, o ambiente utilizado foi uma sala retangular com 360 unidades de largura e 330 de comprimento. As leituras dos sensores foram associadas a variáveis de ponto flutuante e utilizadas diretamente como entrada na rede, sem nenhum pré-processamento. A maior distância que pode ser medida nesta situação corresponde à diagonal do ambiente. Desta maneira, o vetor $\mathbf{D}$ de leituras sensoriais é formado por $[D_0\ D_1\ D_2\ D_3]$, sendo que $D_i \in \mathbb{R}$ e $0 \leq D_i \leq 488,4$.

O robô foi programado para percorrer uma trajetória oval no sentido horário. Os seguintes parâmetros foram utilizados: ρ=0,60, δ=0,10, α=0,002, β=0,9, γ=0,7, I_{max}=100,

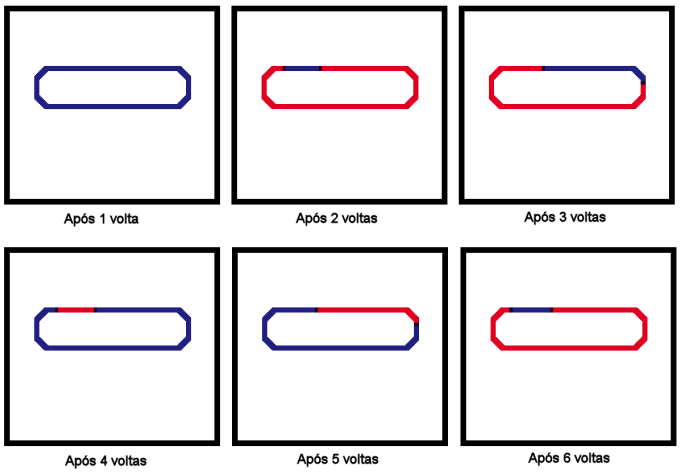

Figura 5.1: Experimento 1 - mapa de disparos

μ=1, η=1 (note que η=1 equivale a desativar a unidade de memória).

A rede permaneceu com os mesmos 2 neurônios existentes em sua inicialização, sem criar nenhum novo. A figura 5.1 mostra qual foi o neurônio vencedor em cada um dos pontos, nas 6 voltas executadas. Estão marcados em azul os locais onde o último neurônio vencedor foi n_1, e em vermelho, n_2.

Há uma questão fundamental a ser discutida neste resultado: por que não foi criado nenhum neurônio novo? Para que um neurônio seja criado, duas condições devem ser satisfeitas: $exp(-||\mathbf{z}(t) - \mathbf{w}_v(t)||) < \rho$ e $w_v^h(t) < \delta$.

A primeira condição é o controle da granularidade da rede. A expressão $exp(-||\mathbf{z}(t) - \mathbf{w}_v^z(t)||)$ é uma medida do erro de quantização, que será chamada de S. Além de utilizada na regra de inclusão de novos neurônios, ela também serve para calcular o estímulo externo aplicado na unidade de habituação. Quando esta medida está próxima de 1, significa que a distância entre o protótipo do neurônio vencedor e o valor de entrada é pequena (erro baixo). Quando a distância aumenta, ela tende a zero. O parâmetro ρ é o limite, abaixo do qual o erro de quantização é considerado inaceitável.

É importante conseguir relacionar o valor de S com a distância entre os vetores do domínio analisado. Tomemos como exemplo um nodo n, com um vetor de pesos $\mathbf{w}$. Seja (x, y) a posição no ambiente onde o vetor D de entradas sensoriais é igual a $\mathbf{w}$. Quando o robô estiver nesta posição, o valor de S para o nodo n é 1. Agora suponhamos que o robô parte da posição (x, y) e avança por um corredor, de maneira que as distâncias laterais permanecem as mesmas, a frontal diminui e a traseira aumenta. Se o vetor $\mathbf{w}$ não se alterar, quantas unidades ele pode caminhar até que $S < \rho$? Ora, se $exp(-d) < \rho$ então $d > ln(\rho)$. Como neste exemplo as distâncias laterais estão fixas e $\rho = 0,60$, então: $d > \frac{\sqrt{-ln(0,6)}}{2} => d > 0,357$. As distâncias envolvidas no experimento analisado são maiores do que isto, então nota-se claramente que não foi esta a condição que impediu a criação de novos nodos.

A segunda condição para inclusão de um neurônio novo é que $w_v^h(t) < \delta$. O objetivo é identificar se este é um padrão novo, para o qual a rede ainda não foi treinada ou se é um padrão antigo. Valores de $w_v^h(t)$ próximos de 0 indicam que o neurônio está habituado (treinado). A expressão que regula o valor de $w^h(t)$ é: $w^h(t) = w^h(t-1)+\gamma(\alpha(1-w^h(t-1)) - \beta w^h(t-1)S(t))$. O mesmo S utilizado como medida do erro de quantização, serve

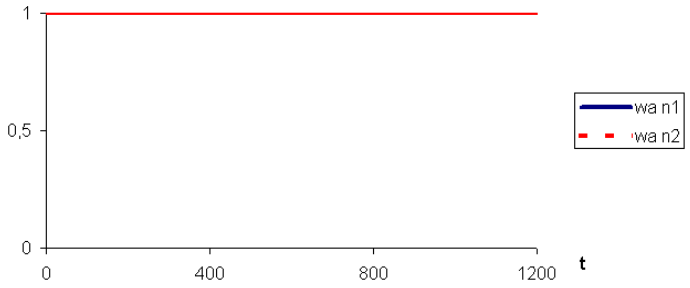

Figura 5.2: Experimento 1 - unidades de habituação

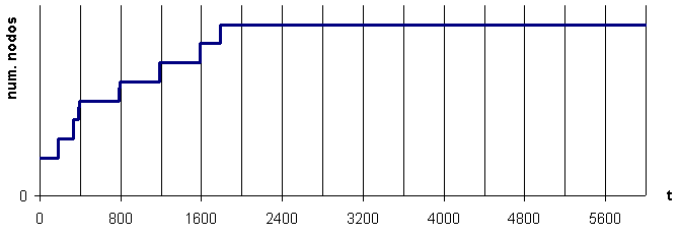

Figura 5.3: Experimento 2 - número de nodos

como estímulo à unidade de habituação. O problema é que, com as distâncias envolvidas neste experimento o valor de S será sempre muito próximo de 0, pois os valores caem na região saturada da curva exponencial. Isto faz com que w^h fique sempre em 1, e impede a criação de novos neurônios. A figura 5.2 mostra o valor da habituação dos dois neurônios no experimento.

Há duas maneiras de solucionar este problema. Uma delas é fazer uma mudança de escala nos vetores de entrada, de maneira que as distâncias fiquem no intervalo [0,1] ou [0,2], por exemplo. Outra alternativa é adicionar um fator de escala E na medida S: $exp(-E||\mathbf{z}(t) - \mathbf{w}_v(t)||)$. Optou-se aqui pela primeira solução.

5.2 Experimento 2

No segundo experimento, utilizou-se ambiente e trajetória semelhante ao primeiro. Na entrada da rede foi utilizado $\mathbf{D}/100$, o vetor com as leituras dos sensores transformado em uma escala 1:100. Os parâmetros utilizados foram: ρ=0,40, δ=0,10, α=0,002, β=0,9, γ=0,7, I_{max}=100, μ=1, η=1.

Após 5 voltas, a rede criou 9 nodos e permaneceu com este número, conforme pode ser visto na figura 5.3. Porém, a análise dos dados mostrou que os pesos dos neurônios não estabilizaram, e que os neurônios mudam, a cada volta, a posição da trajetória na qual disparam.

Tomando um nodo como exemplo, a figura 5.4 mostra as posições da trajetória onde n_3 foi vencedor nas voltas 10 a 15. O número da volta está indicado na figura à direita de cada mapa. Traços verticais foram colocados para facilitar a comparação entre voltas consecutivas. Nota-se claramente que, a cada volta, o campo de disparo avança na mesma direção em que o robô está se deslocando.

Uma característica dos dados de entrada utilizados neste problema é a presença de

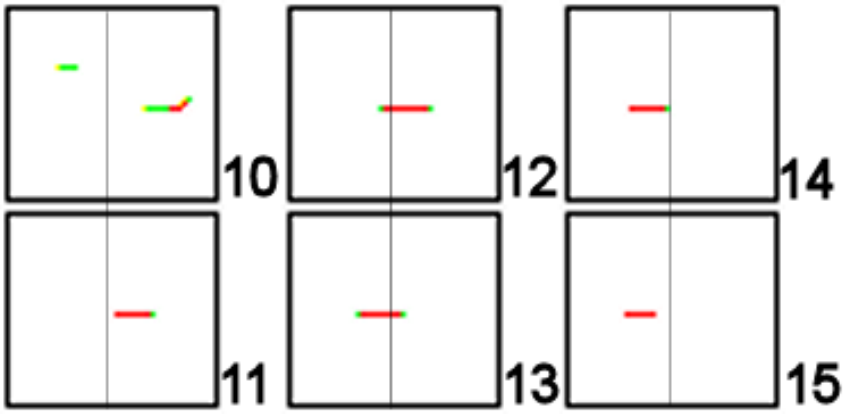

Figura 5.4: Experimento 2 - mapas de disparos do nodo n3

sequências de valores similares, lentamente crescentes ou decrescentes. Por exemplo, quando o robô desloca-se em direção a uma parede, o sensor de distância posicionado à sua frente retornará, a cada leitura, um valor um pouco menor do que o anterior. Os sensores posicionados nos lados retornarão o mesmo valor repetidamente. A consequência disto para o treinamento da rede é que o mesmo neurônio será escolhido o vencedor sucessivas vezes, até que haja uma mudança brusca no padrão de entrada. Isto ocorre porque, a cada iteração o vetor de pesos do neurônio desloca-se na mesma direção do vetor de entrada, pelo processo de ajuste.

Como o neurônio vencedor da rede GTSOM é uma informação que será usada para tomada de decisão, deseja-se que os protótipos da rede modifiquem-se no início (logo após a criação do neurônio), mas depois sofram apenas pequenos ajustes. Caso contrário, o mesmo neurônio poderia ser treinado para detectar uma determinada configuração do agente e posteriormente, através do ajuste de pesos, representar uma situação diferente. Tal mudança traria um efeito negativo no aprendizado. Por isso, o ajuste dos pesos na rede GTSOM foi alterado para a forma mostrada na equação 5.1.

$$\mathbf{w}(t+1) = \mathbf{w}(t) + e\sqrt{w^h(t)}[\mathbf{z}(t) - \mathbf{w}(t)] \tag{5.1}$$

A alteração feita em relação à equação original (ver equação 2.9) foi a inclusão do fator $\sqrt{w^h(t)}$, a raiz da sinapse de habituação do neurônio. O valor da sinapse de habituação se aproxima de 0 cada vez que o neurônio vence a competição, e tende a 1 quando ele fica sem vencer. Desta maneira, evita-se o efeito de arrasto dos pesos, e a representação criada pela rede torna-se estável, permitindo que os protótipos sejam usados como detectores de determinadas situações e utilizados para aprendizado. Os próximos experimentos utilizarão esta versão da regra de ajuste dos pesos.

5.3 Experimento 3

A alteração na regra de ajuste dos pesos permite que os neurônios da rede fiquem fiéis a um determinado padrão de entrada. À medida que o valor da sinapse de habituação do neurônio diminui, os ajustes nos pesos vão ficando menores. Isto cria um outro problema: Caso w^h fique muito pequeno antes que o neurônio consiga ajustar adequadamente os seus

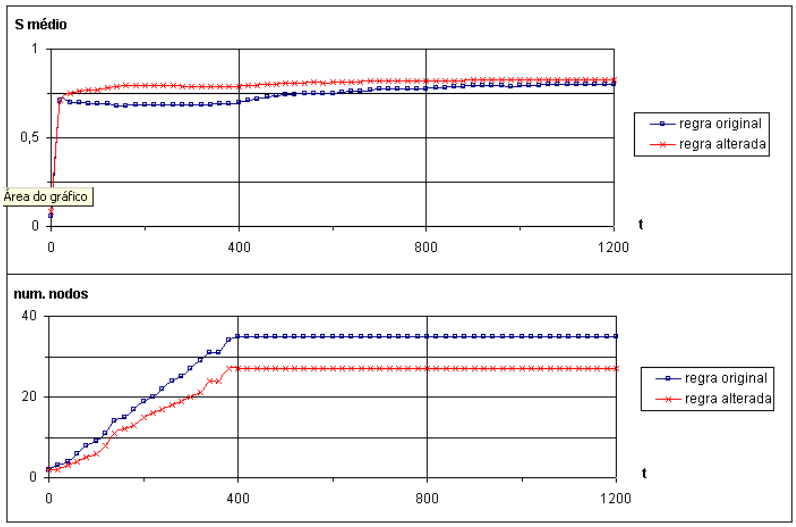

Figura 5.5: Experimento 3 - erro médio

pesos, pode ser criado um novo neurônio desnecessário para o mesmo padrão de entrada. A solução adotada para isto foi mudar o valor inicial dos pesos na criação do neurônio. Ao invés de criá-lo como a interpolação do vetor de pesos da unidade vencedora e do padrão de entrada, o vetor de pesos do neurônio novo é inicializado com o próprio padrão de entrada. Isto permite uma convergência mais rápida.

Para demonstrar a diferença entre os método de inicialização dos pesos, foi repetido o experimento 2 (porém com a regra de ajuste dos pesos alterada). Foi feita uma execução com o método original de inicialização dos pesos e outra com o método alterado. Para medir a diferença entre as duas, foi utilizado o valor médio de S.

A figura 5.5 mostra os resultados. O robô demora 400 unidades de tempo para realizar uma volta, portanto na figura 5.5 estão mostradas as 3 primeiras voltas. O gráfico superior exibe a medida S, enquanto na parte inferior está mostrado o número de neurônios da rede. Nota-se claramente que a utilização da entrada como valor inicial dos pesos produz um melhor resultado neste contexto. Além de conseguir um erro de quantização menor, utiliza menos neurônios para representar os padrões de entrada.

5.4 Experimento 4

Neste experimento, ao invés de percorrer uma trajetória fixa, o agente movimenta-se livremente pelo ambiente com um comportamento de exploração. O objetivo é, novamente, analisar a representação gerada sob o ponto de vista de sua utilização no aprendizado de tarefas.

O ambiente utilizado no experimento possui a forma de uma cruz, conforme mostra a figura 5.6.

Os parâmetros utilizados foram ρ=0,70, δ=0,30, α=0,002, β=0,9, γ=0,7, I_{max}=100, μ=1, η=1. Neste e em todos os experimentos mostrados a seguir neste texto foi utilizada a versão modificada da rede GTSOM, com a mudança nas regras de ajuste e inicialização de pesos descritas nos experimentos 2 e 3.

Após a execução do experimento por 6.000 unidades de tempo, a rede atingiu a forma de um grafo com 21 nodos dispostos em 5 regiões desconexas, conforme a figura 5.7. Note

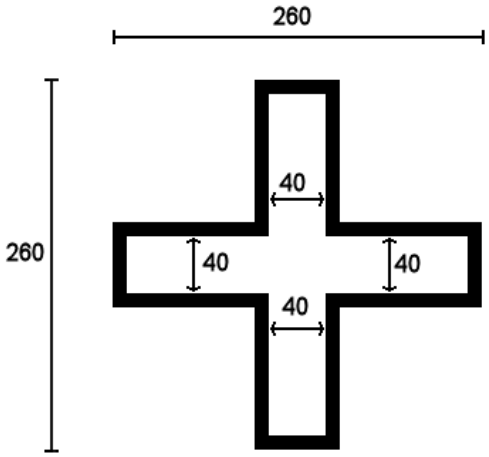

Figura 5.6: Experimento 4 - ambiente

que, como os pesos dos neurônios possuem 4 componentes, não é possível representá-los em um espaço de duas dimensões. Por isso, as posições dos neurônios na figura foram escolhidas de maneira a facilitar a visualização do grafo e não possuem nenhuma relação com os seus pesos, que podem ser consultados na tabela 5.1. É importante lembrar que w_0 é o peso relacionado ao sensor que mede a distância à frente do agente, w_1 à esquerda, w_2 atrás e w_3 à direita.

Uma questão neste resultado que merece atenção especial é a disposição dos neurônios em 5 agrupamentos. Segundo as regras do GTSOM, conexões são criadas entre um neurônio vencedor e o segundo melhor colocado. A idade das conexões é aumentada quando ele participa da vizinhança do vencedor mas não é o segundo. Desta maneira, estimula-se a formação de conexões entre neurônios que representam situações próximas no espaço de entrada. Por isso, espera-se no caso deste experimento que os neurônios vizinhos representem situações semelhantes para o agente. Para verificar isto, vamos examinar os locais de disparo de cada um deles.

A figura 5.8 mostra as posições dentro do ambiente onde o agente se encontrava quando um dos neurônios de cada um dos agrupamentos venceu a competição. Nota-se que cada um dos agrupamentos corresponde a um conceito do domínio representado. Os neurônios dos agrupamentos 1 e 3 representam situações onde o agente está se deslocando ao longo de um corredor. A diferença entre os dois, que não fica perceptível na figura, é que no primeiro caso o agente está deslocando-se em direção ao centro (com uma parede próxima as suas costas), enquanto no segundo ele está se afastando do centro (parede próxima a frente). Isto pode ser percebido pelos pesos mostrados na tabela 5.1. Os agrupamentos 4 e 5 contém neurônios que disparam quando o agente está deslocando-se perpendicularmente a um corredor, e a característica que os distingüe é a presença de uma parede próxima do lado esquerdo (grupo 4) ou direito (grupo 5). Por fim, o agrupamento 2 está associado com a interseção no centro do ambiente.

Embora a representação criada pela rede possua uma informação útil do ponto de vista do aprendizado de tarefas, ela não é suficiente para resolver problemas que envolvam localização espacial. Por ser baseada em leituras de sensores, ela é ambígua. A criação

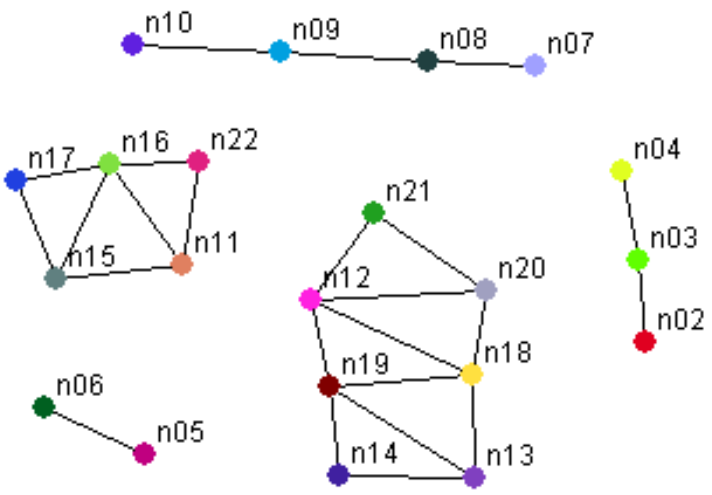

Figura 5.7: Experimento 4 - estado final da rede

Neurônio	w0	w1	w2	w3
n02	2,162	0,244	0,228	0,162
n03	1,862	0,222	0,538	0,178
n04	1,574	0,201	0,826	0,200
n05	1,313	1,182	1,053	1,195
n06	1,111	1,208	1,275	1,192
n07	0,930	0,187	1,470	0,213
n08	0,697	0,174	1,703	0,226
n09	0,405	0,177	1,995	0,223
n10	0,177	0,169	2,223	0,231
n11	0,196	0,320	0,225	2,080
n12	0,190	2,110	0,217	0,290
n13	0,261	1,539	0,139	0,861
n14	0,074	1,531	0,326	0,869
n15	0,281	0,677	0,119	1,723
n16	0,095	0,637	0,305	1,763
n17	0,183	0,898	0,217	1,502
n18	0,316	1,820	0,084	0,580
n19	0,101	1,809	0,299	0,591
n20	0,314	2,334	0,086	0,066
n21	0,062	2,338	0,338	0,062
n22	0,226	0,110	0,174	2,290

Tabela 5.1: Experimento 4 - pesos

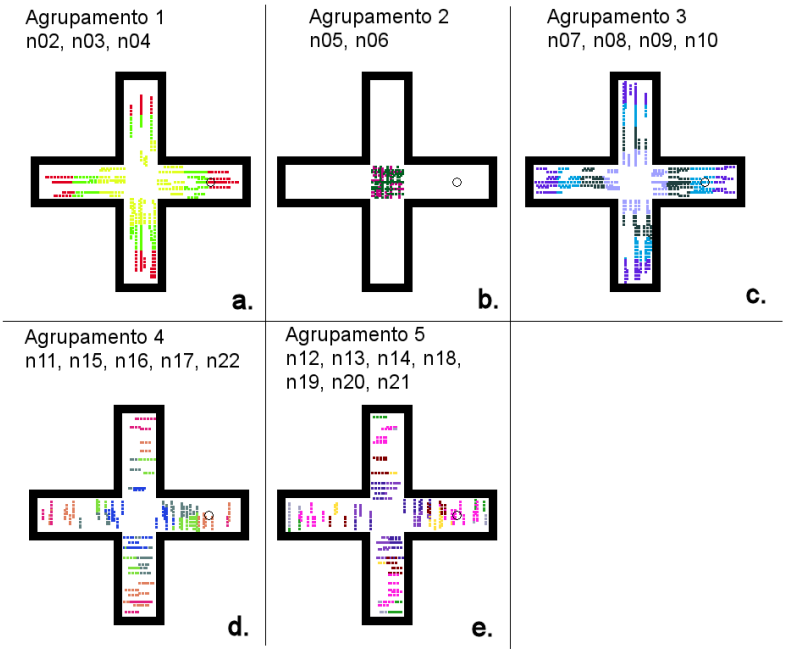

Figura 5.8: Experimento 4 - posições dos disparos

de uma representação de estado que possua a propriedade de Markov requer a eliminação destas ambiguidades. Por isso, optou-se por combinar o mapa perceptivo criado pela rede GTSOM com um mapa espacial, conforme será descrito no capítulo 6.

5.5 Considerações do Capítulo

Os experimentos realizados mostraram a necessidade de realizar adaptações na rede GTSOM para que a mesma pudesse ser utilizada com a finalidade que está sendo proposta, que é a de criar uma representação discreta de estado propícia para o uso em aprendizado por reforço.

Foram propostas alterações na forma de inicialização e ajuste dos pesos. Com essas alterações, a rede consegue se adaptar mais rápido aos padrões de entrada e manter uma representação estável ao longo do tempo. Destaca-se este fato como contribuição importante deste trabalho.

Também ficou evidenciado que a rede GTSOM sozinha não consegue diferenciar entre locais do ambiente que possuem percepções sensoriais semelhantes. Mesmo que fosse utilizado o aspecto temporal, implementado na rede por uma memória de curto prazo, não seria possível suprir essa necessidade.

Para fazer essa diferenciação, uma possível solução seria criar uma estrutura de alto nível considerando o histórico de estados visitados e ações executadas. Desta maneira, dois locais semelhantes no ambiente seriam diferenciados através do caminho que foi percorrido para chegar até lá. Porém, a implementação desta solução não é simples e está fora do alcance deste trabalho.

Foi adotada uma solução mais simples, que consiste em utilizar um mapa de grade em conjunto com a rede GTSOM, para fazer com que estados sensoriais semelhantes ocorrendo em locais diferentes do mapa possam ser considerados diferentes estados para

fins de aprendizado.

6 MODELO PROPOSTO

Neste capítulo será descrito o modelo projetado para atender ao problema de pesquisa. Inicialmente, será fornecida uma visão abrangente do modelo desenvolvido. Em seguida, será descrito o simulador que foi criado para a execução dos experimentos. Logo após, será detalhado cada um dos componentes da arquitetura, seu funcionamento interno e seu papel no sistema.

6.1 Visão Geral do Modelo

O objetivo deste trabalho é projetar uma arquitetura de agente autônomo capaz de criar uma representação do ambiente, utilizando a rede GTSOM, e realizar o aprendizado de tarefas simples em cima desta representação. A figura 6.1 mostra um diagrama geral do modelo desenvolvido com esta finalidade. Os módulos estão organizados em 3 níveis: execução, categorização e planejamento.

No nível de execução estão os componentes que interagem diretamente com o ambiente, recebendo informações através de sensores de distância e de odometria e enviando comandos aos atuadores do robô que realizam o deslocamento.

O nível de categorização é responsável por transformar as leituras dos sensores em uma representação de estado compacta, que possua o mínimo de informação necessária para a tomada de decisão. Isto é feito por 3 componentes: a rede GTSOM, o módulo de mapeamento e o identificador de estados.

O terceiro e último nível, chamado Nível de Planejamento, é onde ocorre o aprendizado por reforço. Utilizando a informação de estado gerada pelo nível de categorização, juntamente com o sinal de reforço, ele seleciona, a partir de um conjunto discreto de ações possíveis, qual a ação a ser tomada a cada instante de tempo.

6.2 Simulador Robótico

Para permitir que os esforços da pesquisa fossem focados na configuração da rede neural e no método de aprendizado, foi criada uma biblioteca de classes em Java, utilizada para simular o funcionamento do robô e sua interação com o ambiente. Procurou-se utilizar sensores e atuadores que são comuns em robôs reais, de maneira a facilitar uma futura implementação física do modelo.

A classe Agente representa um robô, que é dotado de n de sensores de distância distribuídos uniformemente em volta do seu corpo. Um método desta classe realiza a leitura dos sensores, e retorna um vetor contendo a distância até o obstáculo mais próximo em cada uma das direções onde estão colocados os sensores. A movimentação do robô é

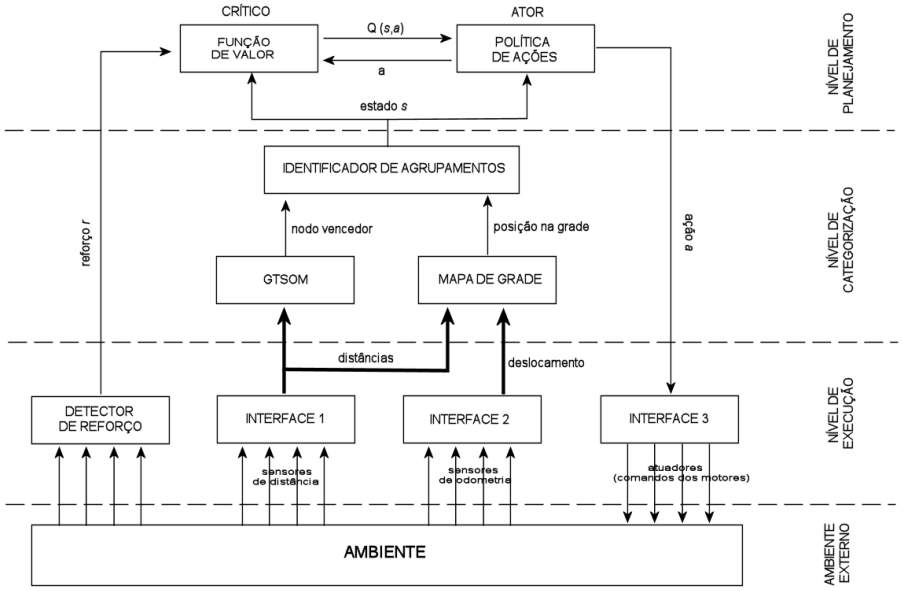

Figura 6.1: Visão geral do modelo

Método	Descrição	Parâmetros	Retorno
Agent	construtor da classe	número de sensores	
readSensors	faz a leitura dos sensores		vetor de distâncias
moveForward	move o robô para frente	número de passos	
turnLeft	gira o robô para a esquerda	ângulo	
turnRight	gira o robô para a direita	ângulo	

Tabela 6.1: Métodos da classe Agente

feita através de atuadores, que fornecem comandos para mover-se para frente, girar para a esquerda ou girar para a direita. A tabela 6.1 resume os principais métodos desta classe que refletem a API de programação do robô.

A criação do ambiente é feita através de outra classe, que fornece mecanismos para que sejam definidos os limites do ambiente (paredes), e indicadas posições especiais onde o agente recebe um sinal de reforço positivo ou negativo, utilizado no aprendizado. O sistema de controle não comunica-se diretamente com esta classe, uma vez que toda a interação com o ambiente é feita pelos sensores e atuadores. Desta maneira, deus detalhes de implementação não são relevantes para compreensão do modelo e não serão descritos aqui.

6.3 Interfaces de Entrada

As interfaces de entrada realizam o pré-processamento dos dados fornecidos pelos sensores do robô. No modelo desenvolvido, as duas grandes fontes de informação do ambiente são:

- Leitura dos sensores de distância: fornece informações que são utilizadas pelo mapa sensorial da rede GTSOM e também pelo módulo de mapeamento;

- Leitura dos odômetros: a informação fornecida pelos odômetros é utilizada pelo processo de mapeamento para atualizar a posição corrente do robô.

A Interface 1, mostrada na figura 6.1, realiza uma mudança de escala nos dados lidos dos sensores de distância. Tal alteração é necessária para o correto funcionamento da rede GTSOM, como ficou evidenciado no experimento descrito na seção 5.1. Seu comportamento pode ser descrito matematicamente como:

$$z(t) = \mathbf{A}d(t) \tag{6.1}$$

onde $d(t)$ é o vetor contendo as leituras dos sensores, $\mathbf{A}$ é a constante de escala e $z(t)$ é o vetor modificado, que é utilizado como entrada da rede GTSOM. Em todos os experimentos descritos neste texto, foi utilizado $\mathbf{A} = 0,1$.

A interface 2 está associada aos sensores de odometria, e realiza o pré-processamento das informações de deslocamento $(\Delta x, \Delta y)$ lidos dos odômetros.

6.4 O Nível de Categorização

A rede GTSOM (BASTOS, 2007) desempenha um papel central no modelo desenvolvido. Sua tarefa é transformar os dados multidimensionais e contínuos lidos dos sensores em uma representação discreta, permitindo o uso de aprendizado por reforço convencional. As características da rede permitem que ela realize aprendizado online, sem fases separadas de treinamento e atuação. Durante a operação, a rede é capaz de criar automaticamente nodos para representar situações que ainda não havia encontrado, e eliminar aqueles que não são mais utilizados.

Os nodos criados pela rede GTSOM podem ser interpretados como estados no espaço de configuração sensorial do robô. Cada estado corresponde a uma situação, como por exemplo: início do corredor, encruzilhada, entrada de uma sala, etc. Isto é uma informação importante, porém não suficiente para a tomada de decisão. Com os sensores utilizados, não é possível representar sem ambigüidade as situações do ambiente. A mesma sensação pode ser observada em locais diferentes (ver exemplo na figura 5.8).

O mapa de grade foi introduzido com o objetivo de eliminar as ambigüidades introduzidas pela representação sensorial. Desta forma, caso existam dois corredores semelhantes, por exemplo, eles serão representados no nível de categorização como estados diferentes devido à sua posição no mapa, mesmo que no GTSOM correspondam ao mesmo nodo.

Pode ser utilizada para o mapa qualquer técnica capaz de realizar auto-localização, combinando as informações dos sensores de distância e de odometria, como por exemplo o filtro de Kalman (MACHADO, 2003). A saída deste módulo tem a forma de coordenadas (x, y) em um mapa bidimensional.

Juntando as informações geradas pela rede GTSOM e pelo módulo de mapeamento, é criada a representação interna do estado do ambiente. O módulo que realiza esta tarefa é chamado de identificador de estados. Ele recebe como entrada uma tupla (n_v, x, y), onde n_v é o nodo vencedor e x e y são as coordenadas no mapa de grade, e fornece como saída o estado s. Sua lógica interna é tal que são criadas subdivisões dentro de cada estado sensorial (nodos da rede GTSOM), cada um relacionado a uma região no ambiente.

A figura 6.2 mostra a estrutura de dados deste módulo. Ele possui uma tabela com os nodos GTSOM. Para cada linha da tabela, há uma lista de estados associada. Os itens da lista armazenam a média $(\overline{x}, \overline{y})$ da posição obtida a partir do mapa de grade, nos instantes de tempo em que o estado foi selecionado.

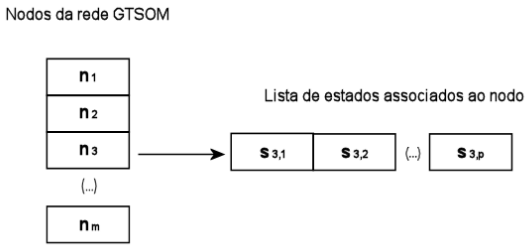

Figura 6.2: Nível de Categorização

Na inicialização, cada nodo da rede GTSOM é associado a uma lista de estados vazia. A cada passo de execução, é percorrida a lista de estados correspondente ao nodo vencedor, buscando um estado cuja posição média $(\overline{x}, \overline{y})$ não fique distante de (x, y) por mais de R unidades. Caso seja encontrado um estado s_i na lista que satisfaça a esta condição, sua posição média $(\overline{x}, \overline{y})$ é atualizada, e o estado s_i é repassado ao nível de planejamento. Se nenhum estado da lista satisfizer a condição, é criado um estado novo, cuja média $(\overline{x}, \overline{y})$ é inicializada com (x, y).

Através do ajuste do parâmetro ρ da rede GTSOM é definida a granularidade da representação no espaço sensorial. Quanto maior a granularidade, mais detalhada é a representação. Porém, uma representação mais detalhada implica em aumento no número de estados, que torna o aprendizado mais demorado no nível de planejamento. Já o parâmetro R do identificador de estados controla a distância necessária para que duas ocorrências do mesmo nodo sejam consideradas como estados diferentes a nível de aprendizado.

6.5 O Nível de Planejamento

O nível de planejamento é responsável por tomar as decisões sobre qual ação executar a cada instante de tempo, utilizando a representação de estado criada pelo nível de categorização, juntamente com o sinal de reforço recebido do ambiente. A forma de aprendizado segue a linha do aprendizado por reforço com ambiente discreto, sendo que o modelo do ambiente não é conhecido pelo agente. O algoritmo utilizado foi o Q-Learning. À medida que o ambiente vai sendo explorado, o agente vai construindo a tabela de valor dos estados, utilizada como base para tomada de decisões.

O aprendizado dá-se de forma discreta, sempre que ocorre uma mudança no estado do ambiente. Nos passos de tempo em que o estado permanece igual ao anterior, não ocorre aprendizado e o agente continua executando a mesma ação. Em outras palavras, uma vez que o agente seleciona o caminho a ser tomado (ex: reto), ele permanece executando a mesma ação até que alguma coisa mude. A figura 6.3 ilustra este processo com um exemplo. Inicialmente, o nível de classificação identifica o estado do ambiente como sendo S_a. O planejador então escolhe a ação a_1. Em $t = 2$, o ambiente continua no mesmo estado, portanto o agente continua executando $a1$. No próximo instante de tempo, o estado ambiente muda para S_b. A recompensa R_1 é utilizada para atualizar o valor da função de valor da ação para o estado S_a, e uma nova ação $a2$ é selecionada para execução.

Para implementar a função de valor das ações, o agente utiliza uma tabela associativa, cuja chave de acesso é o identificador do estado. O conjunto de estados varia ao longo do tempo, enquanto o conjunto de ações é fixo. Desta maneira, cada entrada na tabela possui os valores de $Q(s, a)$ para todas as ações possíveis. Esta tabela inicialmente é vazia, e

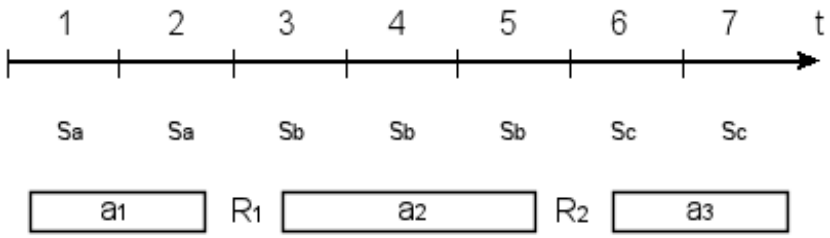

Figura 6.3: Nível de Planejamento. O eixo horizontal indica o tempo. Sa, Sb e Sc são estados identificados pelo nível de classificação, a_1, a_2 e a_3 são ações e R_1 e R_2, recompensas recebidas.

vai sendo preenchida à medida que o agente vai encontrando novos estados. Sempre que, durante a etapa de classificação, algum nodo for removido da rede GTSOM, todos os estados do ambiente associados ao nodo que foi removido são removidos do conjunto de estados.

A cada instante, o agente verifica o estado do ambiente. Enquanto o estado permanece o mesmo, não há nenhum aprendizado e a mesma ação continua sendo executada. Quando o estado muda, a primeira coisa a fazer é verificar se é um estado já conhecido (existente na tabela de estados) ou se é um estado novo. Caso seja um estado novo, ele é inserido na tabela, e os valores $Q(s, a)$ do estado são inicializados em 0. Depois disso, o valor de $Q(s, a)$ do par (estado,ação) anterior é atualizado, utilizando a regra do Q-Learning (equação 6.2), e uma nova ação é selecionada.

$$Q(s_{t-1}, a) \leftarrow Q(s_{t-1}, a) + \alpha[r + \gamma\, max_{a'} Q(s_t, a') - Q(s_{t-1}, a)] \tag{6.2}$$

A política de seleção de ações é ϵ-gulosa. Normalmente seleciona-se a ação que possui o maior valor de $Q(s, a)$ no estado atual, conforme a equação 6.3. Em algumas vezes, uma ação aleatória é selecionada. O parâmetro ϵ regula a frequência com que a ação é selecionada de forma aleatória.

$$a \leftarrow \arg\max_{a'} Q(s_t, a') \tag{6.3}$$

É importante notar que o aprendizado do ambiente (no nível de categorização), ocorre juntamente com o aprendizado das ações (valor dos estados). Ou seja, o nível de planejamento não trabalha com um conjunto de estados pré-definido, uma vez que os estados podem ser criados e removidos durante a operação.

6.6 Algoritmo de funcionamento

Nesta seção será visto de forma abrangente o funcionamento do modelo, envolvendo os diversos componentes. Não existem fases separadas de treinamento e operação, o robô aprende à medida em que vai executando a tarefa. Sua operação se dá na forma de episódios. Ele opera até que atinja o objetivo ou colida com um obstáculo. Quando isso acontece, um novo episódio é iniciado.

O funcionamento do modelo pode ser descrito através dos passos listados no algoritmo 5. $\mathbf{A}$ é o fator de escala, S é o conjunto de estados, α é a taxa de aprendizado, r é a recompensa recebida, γ é a taxa de desconto e ϵ é o fator de exploração.

repita
  `// Realizar a leitura dos dados dos sensores`
  $\mathbf{D}(t) \leftarrow$ dados lidos dos sensores;
  `// Fazer o pré-processamento dos dados sensoriais`
  $\mathbf{z}(t) \leftarrow \mathbf{D} \cdot \mathbf{d}(t)$;
  `// Categorização dos dados sensoriais pela rede GTSOM`
  $n_v(t) \leftarrow$ nodo vencedor;
  $B(t) \leftarrow$ conjunto dos nodos que foram apagados neste passo;
  para cada $n \in B(t)$ **faça**
    remove de S todos os estados associados com n;
  fim
  `// Auto-localização pelo módulo de mapeamento`
  $(x, y) \leftarrow$ posição atual estimada no mapa de grade;
  `// Indentificação do estado utilizando nodo vencedor do GTSOM e a posição no mapa`
  $s_t \leftarrow$ estado atual;
  se $s_t \neq s_{t-1}$ **então**
    se $s_t \notin S$ **então**
      inclui na tabela de estados;
    fim
    `// atualiza Q(s,a) do par estado/ação anterior`
    $Q(s_{t-1}, a) \leftarrow Q(s_{t-1}, a) + \alpha[r + \gamma\, max_{a'} Q(s_t, a') - Q(s_{t-1}, a)]$;
    `// seleciona nova ação`
    $r \leftarrow$ número real aleatório entre 0 e 1;
    se $r < \epsilon$ **então**
      $a \leftarrow$ ação aleatória selecionada de A;
    senão
      $a \leftarrow \arg\max_{a'} Q(s_t, a')$;
    fim
  fim
  Executa ação atualmente selecionada;
até *final do episódio* ;

Algoritmo 5: Algoritmo do modelo proposto

7 RESULTADOS EXPERIMENTAIS

Neste capítulo, serão mostrados os resultados obtidos com a aplicação do modelo proposto em alguns experimentos. A tarefa a ser resolvida consiste em uma espécie de labirinto. O agente não possui *a priori* nenhum mapa do ambiente que está sendo explorado, e deve encontrar um objeto cuja localização não é conhecida. Supõe-se que, ao chegar suficientemente perto, o agente é capaz de perceber a presença do objeto e reconhecê-lo.

São três as ações que podem ser utilizadas pelo agente para deslocar-se no ambiente. A cada momento, ele deve escolher entre: virar 90º à esquerda, virar 90º à direita ou continuar se deslocando na mesma direção. Não existe nenhum conhecimento prévio do domínio, de maneira que ele deve aprender a evitar colisões ao mesmo tempo que aprende o caminho.

Os testes do capítulo 5 já avaliaram a natureza da representação do espaço sensorial criada pela rede GTSOM. Agora, deseja-se verificar se a combinação desta rede com um mapa de grade é capaz de segmentar o espaço de estados de maneira a permitir o aprendizado do caminho no labirinto.

7.1 Experimento 1

Neste experimento, será mostrada a capacidade de aprendizado do modelo em um labirinto em forma de cruz. O agente deve percorrer o ambiente mostrado na figura 7.1, partindo da posição indicada com um X e encontrar um objeto que se encontra na área marcada com um círculo azul.

A cada vez que o agente se desloca, recebe um reforço de -1. Quando colide com uma parede, recebe um reforço de -100, e volta para a posição inicial, começando um novo episódio. Se conseguir atingir o objetivo, recebe um reforço de 100 e encerra o episódio.

Os seguintes parâmetros foram utilizados na rede GTSOM: ρ=0,70, δ=0,30, α=0,002, β=0,9, γ=0,7, I_{max}=100, μ=1, η=1. No Q-Learning, foram utilizados α=0,9, γ=0,95, ϵ=0,05.

Foram executados 150 episódios. Uma maneira de avaliar o desempenho na execução da tarefa é medindo o reforço total recebido em cada um deles. Este resultado está mostrado na figura 7.2. Cada ponto corresponde a um episódio, com o reforço no eixo vertical. A linha de tendência é uma média móvel de período 12.

Um aspecto curioso deste resultado é que nos primeiros 50 episódios, a tendência é de redução do reforço recebido, como se o agente estivesse piorando seu desempenho. Na verdade, o que acontece é que eles está aprendendo a evitar colisões. Por isso, aumenta a duração do episódio. Como ele recebe um reforço negativo por cada deslocamento, isso faz com que o reforço total recebido no episódio diminua.

Por volta do 50º episódio, ele consegue aprender a trajetória correta. Neste ponto,

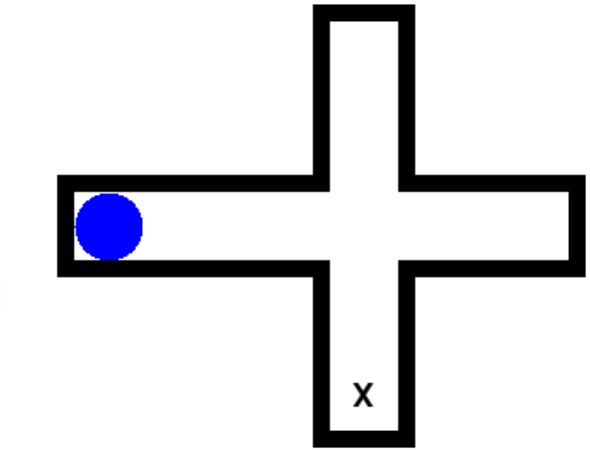

Figura 7.1: Experimento 1 - configuração do ambiente

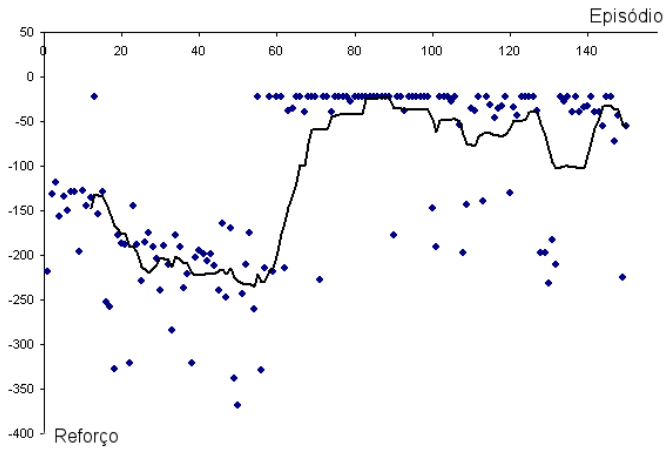

Figura 7.2: Experimento 1 - reforço recebido

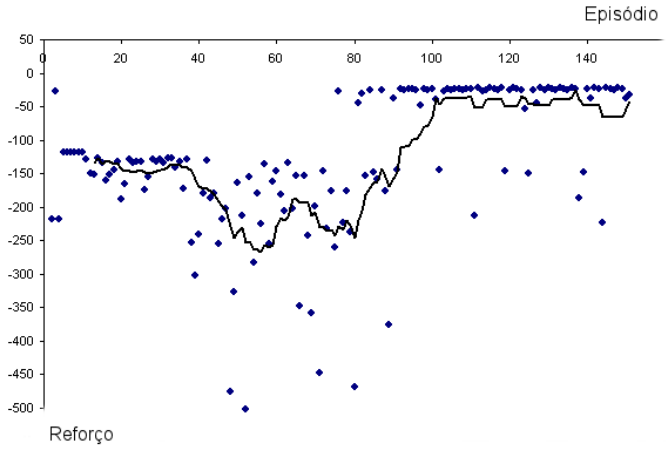

Figura 7.3: Experimento 2 - reforço recebido

ele passa a atingir o objetivo. Ainda continuam ocorrendo colisões em alguns episódios, devido à taxa de exploração ϵ=0,05 associada à política de ações.

7.2 Experimento 2

Neste teste, foi utilizado um ambiente semelhante ao do experimento 1. A diferença é que a posição inicial do agente muda a cada episódio. Esta é uma diferença importante, pois, no experimento 1, bastaria o agente aprender a sequência de ações necessárias para o atingimento do objetivo. Já neste caso onde a posição inicial muda, ele deve ser capaz de localizar-se para conseguir realizar a tarefa com sucesso.

A posição inicial do agente é escolhida entre 3 possibilidades: no final de cada um dos corredores, excetuando-se aquele onde se encontra o alvo. A figura 7.3 mostra os resultados.

Nota-se que a convergência demora um pouco mais do que no experimento 1, em razão do aumento do grau de dificuldade. Porém, uma vez que aprendeu a classificação dos estados e a função de valor, o agente consegue repetir resultados bem sucedidos em quase todos os episódios.

7.3 Discussão

Em todos os experimentos realizados, o agente conseguiu construir de maneira consistente a representação do espaço sensorial, associando-a com as posições no mapa de grade. Com a experimentação, a tabela de valor dos estados vai sendo melhorada até que permita o atingimento do alvo.

Um dos problemas que fica visível quando observamos as trajetórias realizadas pelo agente durante o processo de aprendizado é que a estratégia de exploração não consegue evitar que ele fique repetidamente visitando lugares já conhecidos e demore a encontrar o alvo. Isso acontece porque, embora possa ser dada preferência a ações ainda não executadas, várias ações diferentes podem levar o agente ao mesmo local.

A outra limitação que pode ser observada diz respeito ao método de aprendizagem.

Como o Q-Learning atualiza, a cada iteração, somente o último estado visitado, é necessário que o objetivo seja atingido várias vezes até que a informação do melhor caminho a ser seguido seja propagada até o estado inicial. Isto torna a convergência demorada. Uma possível solução seria a utilização de um método com traço de ativação, que atualize todos os estados que foram visitados. Porém, no modelo utilizado, isto acarretaria em um problema de consistência, pois estados anteriormente visitados podem ter sido removidos.

8 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, foi desenvolvido um modelo de agente robótico que cria uma representação do ambiente reunindo aspectos sensoriais e espaciais. Utilizando esta representação como espaço de estados, o agente é capaz de aprender o caminho em um labirinto simples utilizando técnicas de aprendizado por reforço.

Os resultados obtidos nos testes mostraram que o modelo consegue segmentar adequadamente o espaço de estados, e realiza o aprendizado da tarefa de forma rápida e confiável. O agente consegue aprender a evitar colisões e memorizar a localização do alvo, podendo chegar até ele independentemente de sua posição inicial.

Além de realizar o aprendizado da tarefa, o modelo é capaz de expandir sua representação sempre que se depara com situações não conhecidas, ao mesmo tempo que gradualmente remove da memória aquelas experiências que não se repetem.

O ponto falho do modelo é que a estratégia de exploração é genérica, e não leva em conta as propriedades espaciais do ambiente. Isto traz duas desvantagens: o ocasional deslocamento em circuitos fechados, retornando ao mesmo lugar de partida; e a falta de estímulo a visitar regiões ainda não exploradas, fazendo com que o agente concentre-se nas regiões do espaço de estados que são mais conhecidas.

Parte importante da contribuição deste trabalho encontra-se na avaliação e adaptação da rede GTSOM. Os experimentos relatados mostraram que são necessárias adaptações em seu algoritmo de funcionamento para que ela se torne adequada à utilização como representação de estados para aprendizado. Isto acontece porque, em sua forma original, o algoritmo permite que neurônios treinados a reconhecer um determinado padrão de entrada alterem seus pesos de maneira significativa, passando a representar uma situação diferente. Isto é prejudicial quando os neurônios são vistos como estados markovianos, pois causaria instabilidade na função de valor de estados. Porém, após as adaptações realizadas, a rede mostrou-se suficiente e adequada para a tarefa.

O simulador desenvolvido durante o estudo permitiu que o esforço fosse focado na avaliação da rede GTSOM e no desenvolvimento da arquitetura de aprendizado, sem que fosse necessário preocupar-se com detalhes de implementação, o que fatalmente ocorreria caso o trabalho fosse desenvolvido utilizando um robô real, ou mesmo um simulador fornecido pelo fabricante. Seria importante, para validação deste modelo, a implementação do modelo projetado em um robô real. Uma vez que os sensores utilizados estão disponíveis na grande maioria dos robôs comerciais, e o algoritmo não realiza processamento de dados intenso, tal implementação é direta e requer apenas tempo para desenvolvimento. Desta maneira, poderia-se verificar de que maneira o aprendizado ocorre na presença de ruídos nos sensores e imprecisão na execução das ações. Tal tarefa é sugerida como trabalho futuro.

Outra questão que fica em aberto diz respeito à utilização da informação contida nas

conexões criadas pela rede GTSOM. Somente a informação do nodo vencedor está sendo utilizada no modelo. Mas, como foi mostrado nos experimentos, a rede de conexões dá informações sobre nodos que possuem propriedades semelhantes, como corredores e encruzilhadas. Acredita-se que tal informação possa ser utilizada de alguma forma para agilizar o aprendizado, porém não está claro de que maneira encaixá-la no modelo.

Outras melhorias sugeridas como trabalhos futuros são: a utilização de uma estratégia de exploração que favoreça ações que levam a lugares ainda não visitados, e a investigação de como um algoritmo de aprendizado que utilize traço de ativação pode ser utilizado de maneira consistente mesmo com a possibilidade de remoção de estados visitados.

REFERÊNCIAS

BASTOS, E. N. F. **Uma Proposta de Rede Neural Auto-Organizável, Temporal e Construtiva Voltada a Aplicações Robóticas**. 2007. Dissertação (Mestrado em Ciência da Computação) — UFRGS.

BERGLUND, E.; SITTE, J. The Parameterless Self-Organizing Map Algorithm. **IEEE Transactions on Neural Networks**, [S.l.], v.17, n.2, p.305–316, 2006.

BLACKMORE, J. **Visualizing High-Dimensional Structure with Incremental Grid Growing Neural Network**. 1995. Dissertação (Mestrado em Ciência da Computação) — University of Texas at Austin, Austin.

BROWNING, B. **Biologically Plausible Spatial Navigation for a Mobile Robot**. 2000. Tese (Doutorado em Ciência da Computação) — The University of Queensland.

BURZEVSKI, V.; MOHAN, C. K. Hierarquical Growing Cell Structures. In: INTERNATIONAL CONFERENCE ON NEURAL NETWORKS, 1996. **Proceedings...** [S.l.: s.n.], 1996. v.3, p.1658–1663.

CHAPPEL, G.; TAYLOR, J. G. The Temporal Kohonen Map. **Neural Networks**, [S.l.], v.6, n.3, p.441–441, 1993.

CHONG, K. S.; KLEEMAN, L. Feature-Based Mapping in Real, Large Scale Environments Using an Ultrasonic Array. **The International Journal of Robotics Research**, [S.l.], v.18, n.1, p.3–19, 1999.

DOYA, K.; SAMEJIMA, K.; ICHI, K.; KAWATO, M. **Multiple Model-based Reinforcement Learning**. 2000.

FRITZKE, B. Growing Cell Sctructures—a Self-Organizing Network in k Dimensions. In: ALEKSANDER, I.; TAYLOR, J. (Ed.). **Artificial Neural Networks, 2**. Amsterdam, Netherlands: North-Holland, 1992. v.2, p.1051–1056.

FRITZKE, B. Growing Grid—a self-organizing network with constant neighbourhood range and adaptation strength. **Neural Processing Letters**, [S.l.], v.2, n.5, p.9–13, 1995.

FRITZKE, B. A growing neural gas network learns topologies. In: TESAURO, G.; TOURETZKY, D. S.; LEEN, T. K. (Ed.). **Advances in Neural Information Processing Systems 7**. Cambridge MA: MIT Press, 1995. p.625–632.

HAYKIN, S. **Redes Neurais**: princípios e prática. 2.ed. Porto Alegre: Bookman, 2001.

HODGE, V. J.; AUSTIN, J. Hierarquical Growing Cell Structures: TreeGCS. **IEEE Transactions no Knowledge and Data Engineering**, [S.l.], v.13, n.2, p.207–218, Mar. 2001.

JAMES, D. L.; MIIKKULAINEN, R. SARDNET: a self-organizing feature map for sequences. In: TESAURO, G.; TOURETZKY, D.; LEEN, T. (Ed.). **Advances in Neural Information Processing Systems 7**. Cambridge, MA, USA: MIT Press, 1995. p.577–584.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. P. Reinforcement Learning: a survey. **Journal of Artificial Intelligence Research**, [S.l.], v.4, p.237–285, 1996.

KANGAS, J. Time-Delayed Self-Organizing Maps. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL NEURAL NETWORKS, 1990, Berlin, Alemanha. **Proceedings...** [S.l.: s.n.], 1990. v.2, p.331–336.

KOHONEN, T. Self-organized formation of topologically correct feature maps. **Biological Cybernetics**, [S.l.], n.43, p.59–69, 1982.

KUIPERS, B. J.; BYUN, Y.-T. A robust qualitative method for robot spatial learning. In: SEVENTH NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, 1988. **Proceedings...** [S.l.: s.n.], 1988. p.774–779.

LI, G.; PANG, J. A Reinforcement Learning with Adaptive State Space Construction for Mobile Robot Navigation. In: NETWORKING, SENSING AND CONTROL 2006, 2006. **Anais...** [S.l.: s.n.], 2006.

LINAKER, F.; NIKLASSON, L. Time series segmentation using an adaptive resource allocation vector quantization network based on change detection. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS, 2000, Washington DC, USA. **Proceedings...** [S.l.: s.n.], 2000. v.6, p.323–328.

MACHADO, K. F. **Módulo de Auto-Localização para um Agente Exploratório usando Filtro de Kalman**. 2003. Dissertação (Mestrado em Ciência da Computação) — UFRGS.

MORAVEC, H. P.; ELFES, A. High Resolution Maps from Wide Angle Sonar. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 1985, addr. **Proceedings...** [S.l.: s.n.], 1985.

MOSER, L. D. **Modelo de um Neurônio Diferenciador-Integrador para Representação Temporal em Arquiteturas Conexionistas.** 2004. Dissertação (Mestrado em Ciência da Computação) — UFRGS.

OWEN, C.; NEHMZOW, U. Landmark-based navigation for a mobile robot. In: FROM ANIMALS TO ANIMATS 5: PROCEEDINGS OF THE 5TH INTERNATIONAL CONFERENCE ON SIMULATION OF ADAPTIVE BEHAVIOUR, 1998, Cambridge. **Anais...** MIT Press, 1998. p.240–245.

SHATKAY, H.; KAELBLING, L. Learning topological maps with weak local odometric information. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE (IJCAI'97), 1997, Nagóia, Japão. **Proceedings...** [S.l.: s.n.], 1997. p.920–992.

SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning**: an introduction. 2.ed. London, England: MIT Press, 1998.

THRUN, S. Learning Metric-Topological Maps for Indoor Mobile Robot Navigation. **Artificial Intelligence**, [S.l.], v.99, n.1, p.21–71, 1998.

THRUN, S. Probabilistic Algorithms in Robotics. **AI Magazine**, [S.l.], v.21, n.4, p.93–109, 2000.

VARSTA, M.; HEIKKONEN, J.; MILLáN, J. D. R. Recurrent Self-Organizing Map for Temporal Sequence Processing. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL NEURAL NETWORKS, 7., 1997, Lausanne, Switzerland. **Proceedings...** [S.l.: s.n.], 1997. p.421–426.

VOEGTLIN, T. Context Quantization and Contextual Self-Organizing Maps. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS, 2000. **Proceedings...** [S.l.: s.n.], 2000. v.5, p.20–25.

WANG, D. **Habituation**. Cambridge, MA, USA: MIT Press, 1998. 441–444p.