Proceedings of the IV Brazilian Conference on Neural Networks - IV Congresso Brasileiro de Redes Neurais
pp. 361-366, July 20-22, 1999 - ITA, São José dos Campos - SP - Brazil

Treinamento Condicionado Através de Esquemas de Transição

Policarpo B. Uliana, Rui Seara, José Carlos M. Bermudez
Universidade Federal de Santa Catarina
Departamento de Engenharia Elétrica
E-mail: poli@linse.ufsc.br

Abstract

This paper proposes a new architecture for reinforcement learning of agents that are controlled via reward or punish stimuli. This proposal is based on transition schemes, which map the agent environment for controlling it. Such mapping uses cause and effect basic relations, which can be used to build complex control structures. This approach is an interesting alternative, leading to a new program strategy, which requires little information, as compared to conventional programming techniques.

1. Introdução

O treinamento condicionado (*reinforcement learning*) é um campo de pesquisa da área de *Inteligência Artificial* (IA) que vem crescendo significativamente nos últimos anos. O objetivo desse tipo de treinamento é a programação de agentes através de um processo de condicionamento, controlado por estímulos de recompensa ou punição, sem que haja necessidade de programação específica.

Os vários modelos de treinamento condicionado discutidos na literatura, tais como Q-*learn* [1], *adaptive heuristic critic* [2], TD(λ) [3], *associative reinforcement comparison* (ARC) [4], *complementary reinforcement backpropagation* (CRBP) [5], ainda não têm resolvido complemente todos os principais problemas relacionados a este tipo de treinamento. Existem grandes obstáculos para se alcançar a meta do treinamento de agentes *artificiais* para a realização de tarefas complexas através deste tipo de condicionamento.

Este artigo propõe um novo tipo de arquitetura baseada em esquemas de transição. Esses esquemas mapeiam o ambiente no qual o agente está inserido, através de relações elementares de causa e efeito que são organizadas de forma hierárquica, permitindo a implementação de estruturas de controle relativamente complexas. Esta proposta representa uma interessante alternativa aos modelos existentes, levando a uma nova estratégia de programação que necessita de reduzida quantidade de informação, quando comparada às técnicas de programação convencionais.

2. Treinamento com Reforço Condicionado

De forma geral, percebe-se que todos os seres vivos são capazes de adaptar-se a um meio ambiente, basicamente, através de dois tipos de comportamentos: instintivos e condicionados.

O treinamento condicionado é um processo em que um agente, inserido em um ambiente, deve descobrir soluções através da aprendizagem por tentativa e erro. O agente recebe uma série de informações do ambiente (inclusive os estímulos de reforço) e, simultaneamente, interage com o ambiente (Fig. 1). Existem, basicamente, dois tipos de reforço: reforço imediato e com retardo. No reforço imediato, o estímulo (positivo ou negativo) é recebido após a realização de cada ação. No reforço com retardo, o estímulo é recebido após a realização de um conjunto de ações.

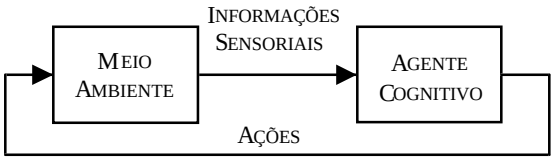

Figura 1: Representação de um agente cognitivo inserido no meio ambiente

O agente pode ser modelado através de uma função de controle que, para um dado estado de entrada, gera um ou mais estados de saída. Desta forma, o agente deve realizar um mapeamento entrada-saída que leve em conta os estímulos de reforço e indique qual a ação de maior *ganho* (a que tenha a maior possibilidade de gerar um estímulo positivo ou evitar um negativo) para cada estado de entrada. Segundo [6], existem duas estratégias principais para a determinação deste mapeamento. A primeira consiste na realização de uma busca exaustiva no ambiente a fim de determinar as ações que apresentem melhor desempenho para cada estado de entrada. Esta busca pode ser realizada através de algoritmos clássicos de otimização, ou através de outros algoritmos, por exemplo, algoritmos genéticos. A segunda estratégia passa pelo uso de técnicas estatísticas e por métodos que permitem estimar a importância de determinadas ações em função de estados do ambiente, que vão sendo gradativamente explorados pelo agente. No treinamento condicionado,

a segunda estratégia é a mais interessante, pois, normalmente, o agente inserido no ambiente passa por um processo de aprendizagem contínuo, no qual não é possível realizar uma busca exaustiva de todos os estados de entrada.

O problema de treinamento condicionado de um agente pode ser descrito através do modelo *Markov decision processes* (MDP) [7], que consiste de:

- um conjunto de estados de entrada (S);
- um conjunto de ações (A);
- uma função de ganho R: $S \times A \rightarrow \Re$;
- uma função de transição de estado T: $S \times A \rightarrow \Pi(S)$, na qual cada elemento de $\Pi(S)$ é a distribuição de probabilidade sobre o conjunto S (ou seja $\Pi(S)$ mapeia a probabilidade de transição dos estados). Pode-se escrever $T(s, a, s')$ como uma função de probabilidade de transição do estado s para o s' quando se executa a ação a.

Obs.: Sem perda de generalidade, esta definição também pode ser aplicada a espaços contínuos.

Um ambiente é dito MDP quando T não depende de qualquer estado ou ação anterior do agente. Um ambiente não-MDP ocorre quando o espaço de entrada é apenas parcialmente observável [6]. Para ambientes não-MDP, podem ser definidas arquiteturas que estimem os estados não-observados com base nos estados e nas ações passadas do agente.

Uma análise ampla dos modelos disponíveis para realização de treinamento condicionado, como a realizada em [6], mostra-nos uma base comum a todos os algoritmos:

- um mecanismo de exploração, no qual a ação A é definida com o objetivo de explorar as funções R e T;
- um mecanismo de escolha, no qual a ação A é definida em função do ganho R e do estado S atual.

Em princípio, a função T não precisa ser conhecida pelo agente, mas se torna importante para o caso de ambientes que gerem reforços com atraso, pois permite a propagação de um reforço recebido em um dado tempo para os estados anteriores.

Os vários modelos de treinamento condicionado existentes diferem, basicamente, quanto aos seguintes aspectos [6]:

- estratégias de otimização do uso dos mecanismos de exploração;
- formas de armazenamento e cálculo estatístico da função R;
- formas de utilização da função T;
- formas de propagação de reforços com atraso;
- formas de implementação (simbólica ou conexionista)
- estratégias de cálculo estatístico que permitam a evolução das funções R e T, quando as mesmas variam com o tempo.

De modo geral, todos estes modelos levam a uma função representada por:

$$a = F(s), \qquad (1)$$

onde F é uma função que leva a uma saída a, para uma dada entrada s, comandada pela função de reforço R.

Para espaço com elevadas dimensões, pode-se subdividir F em um conjunto de funções que mapeiem subespaços de S para saídas intermediárias, que são combinadas para gerar o espaço de A, como mostrado pela Fig. 2.

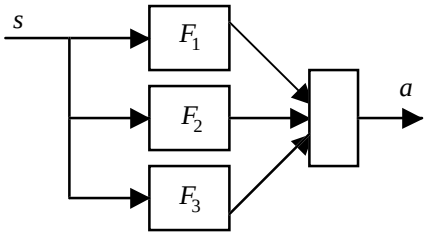

Figura 2: Modelo de controle hierárquico.

A arquitetura proposta rompe este paradigma tradicional de representação de sistemas dado pela equação (1), ou seja:

Entrada → *Processamento* → *Saída*

No novo paradigma, os espaços de entrada e saída são subdivididos em um conjunto de subespaços, sendo que um subespaço de entrada é tratado de forma idêntica a um de saída. O modelo proposto não processa sinais na sua forma convencional, mas sim, unidades de informação que definem determinados *objetivos*. Esses objetivos são caracterizados por conjuntos de estados, que os subespaços de entrada (ou saída) devem assumir ou evitar. Os objetivos podem ser propagados através de uma estrutura hierárquica (Fig. 5). Os diversos subespaços são conectados por relações de causa e efeito que fornecem uma estrutura para propagação de objetivos entre quaisquer subespaços.

3. Modelos de Esquemas

O conceito de esquemas foi introduzido por Piaget na década de 50 [8], através de uma modelagem da capacidade cognitiva de um agente, dada por uma coleção de *esquemas de assimilação*, assim definidos:

- objetos e situações aos quais o esquema se aplica;
- ações a serem tomadas;
- objetivos ou resultados previstos;
- parâmetros de avaliação.

Desde então, alguns autores (Balkenius [9] e Sun [10], dentre outros) propuseram modelos de esquemas matematicamente mais elaborados, porém não tão abrangentes. Recentemente, Drescher [11] e Wazlawick [12] propuseram outros modelos, buscando formalizar a idéia original de esquemas de assimilação, apresentada por Piaget [8].

É interessante notar que existe uma semelhança entre os esquemas propostos por Piaget e o modelo MDP de Markov. Os dois modelos são diferentes no sentido em que o modelo de Markov é voltado para a

descrição de um ambiente. Desta forma, a função $T(s,a,s')$ é válida para todos os estados de entrada e saída e tem um grau de probabilidade fixo associado a cada transição. Já o conjunto de esquemas representa apenas um modelo interno do ambiente, portanto imperfeito, que o agente criou. Os esquemas podem ser também modificados, especializados, generalizados, criados e destruídos, o que não acontece com o MDP. Além disso, dois (ou mais) esquemas podem ser associados (em série ou paralelo) a fim de criar metaesquemas, os quais podem, por sua vez, ser novamente associados, criando uma hierarquia de metaesquemas.

4. Modelo Proposto

O presente trabalho está baseado em uma generalização do conceito de esquemas de assimilação e também do MDP, que denominamos *esquemas de transição* (*ET*).

Definição 1 – Esquema de Transição: Dados dois subespaços quaisquer (de entrada ou saída), um deles é especificado como *espaço de causa* $C = \{c_1, c_2, ..., c_n\}$ e o outro, como *espaço de efeito* $E = \{e_1, e_2, ..., e_m\}$. Assim, um *esquema de transição* é definido pela seguinte função:

$$ET(c_i, c_f, e_i, e_f, t, p, r), \qquad (2)$$

onde:

c_i é o estado inicial do espaço de causa;

c_f é o estado final do espaço de causa;

e_i é o estado inicial do espaço de efeito;

e_f é o estado final do espaço de efeito;

t é o tempo médio de transição;

p é a possibilidade de transição;

r é o parâmetro de avaliação do esquema.

Um esquema de transição é criado quando uma mudança de estado ocorre em *C* e, a seguir (após um tempo *t*), uma mudança de estado é observada em *E*. O parâmetro *p* é calculado através da observação das transições em *C* que levam a uma mesma transição em *E*. O valor do parâmetro *r* é derivado diretamente do sinal de reforço.

A Fig. 3 apresenta uma ilustração deste conceito para um espaço bidimensional, onde os estados são representados por pontos e as transições por flechas.

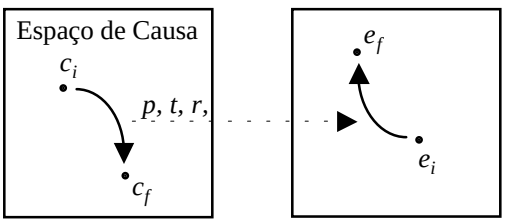

Figura 3: Representação de um esquema de transição.

A) Metaesquemas de transição

O conjunto de todos os esquemas de transição, atribuídos a dois subespaços (*C*, *E*), define um metaesquema de transição (MET). Além das entradas correspondentes aos sinais desses subespaços, um metaesquema de transição possui também uma entrada e uma saída assim definidas:

E_e é a entrada objetivo para o espaço de efeito (de mesma dimensão que *E*);

S_c é a saída objetivo para o espaço de causa (de mesma dimensão que *C*).

O metaesquema, ao receber uma entrada objetivo (E_e) válida, utiliza as informações de todos os esquemas de transição por ele observados para gerar uma saída objetivo (S_c). A Fig. 4 ilustra as entradas e saídas de um metaesquema de transição.

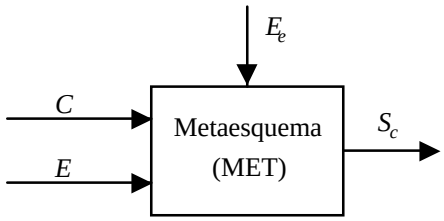

Figura 4: Entradas e saídas de um metaesquema de transição.

Os metaesquemas de transição podem ser interligados de forma hierárquica, o que permite a definição de sistemas de controle relativamente complexos a partir de blocos mais simples, como ilustrado pela Fig. 5.

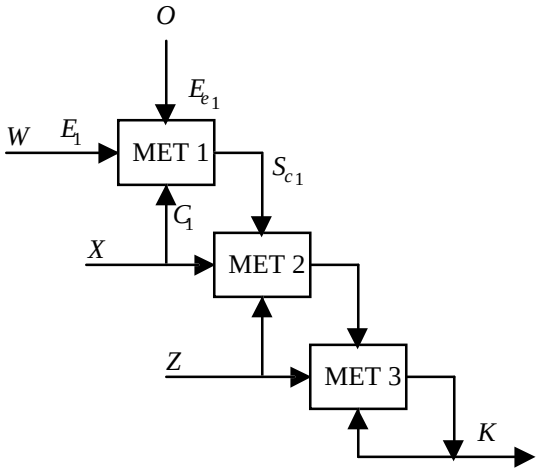

Figura 5: Exemplo de hierarquia de metaesquemas

B) Controle de um metaesquema de transição

Um metaesquema de transição é controlado por uma série de procedimentos:

- manutenção dos esquemas;
- definição de saídas objetivo;
- mecanismo de exploração.

O procedimento de manutenção de esquemas é responsável pela criação e armazenamento de todos os esquemas de transição presentes no metaesquema. Inicialmente, o metaesquema não possui nenhum esquema de transição, e os esquemas vão sendo gerados e atualizados através da observação das mudanças de estado que ocorrem nos espaços *C* e *E*.

O procedimento de definição de saídas objetivo utiliza o sinal E_e para gerar um sinal S_c, definido a partir dos esquemas de transição armazenados. Os objetivos são propagados por uma função gradiente dada pelas possibilidades de transição no espaço *E*.

O mecanismo de exploração gera transições nos diversos espaços de causa, que permitem a criação dos esquemas de transição. Esta exploração pode ser realizada de duas formas distintas:

- utilização de um mecanismo de exploração interno ao agente, que ative suas saídas de forma aleatória ou sistemática (reflete um processo de aprendizagem por tentativa e erro);
- utilização de um mecanismo de controle das saídas externo ao agente (treinamento supervisionado).

C) *Classificação dos esquemas de transição*

Os esquemas de transição podem ser classificados em três grupos:

- esquema de transição completo: $c_i \neq c_f$ e $e_i \neq e_f$;
- esquema de transição sem efeito observável: $c_i \neq c_f$ e $e_i = e_f$ (para t maior do que um tempo máximo);
- esquema de transição sem causa observável: $c_i = c_f$ e $e_i \neq e_f$;

Se não existirem ligações efetivas de causa e efeito entre os subespaços *C* e *E*, poucos esquemas de transição completos serão observados e suas possibilidades serão de baixos valores.

Os esquemas de transição completos são os mais importantes, pois definem os gradientes para a propagação dos objetivos. As transições dos esquemas completos também podem ser classificadas em três grupos:

- transições de causa divergente: apenas o ponto inicial da transição no espaço de causa é importante;
- transições de causa convergente: apenas o ponto final da transição no espaço de causa é importante;
- transições completas: os dois pontos do espaço de causa são importantes.

Esta classificação deve ser levada em conta pelos procedimentos de controle do metaesquema que, através dela, podem:

- não diferenciar os esquemas quando um dos estados de causa é não-significativo;
- definir como saída objetivo somente os estados de *C* que forem relevantes.

D) *Implementação de estruturas baseadas em metaesquemas de transição*

Os metaesquemas de transição podem ser implementados através de um sistema de processamento simbólico, com uma base de dados única, controlada por rotinas executadas seqüencialmente. Considerando que a arquitetura proposta é preponderantemente distribuída, os esquemas de transição também podem ser implementados através de redes neurais.

No Apêndice I, é mostrado um exemplo simples de aplicação, usando a abordagem proposta para o controle de um agente autônomo.

No Apêndice II, são apresentados exemplos de pseudo-códigos para rotinas de controle de metaesquemas de transição.

5. Resultados Experimentais

Os autores aplicaram o treinamento condicionado de agentes, usando metaesquemas de transição, em uma série de problemas simples a fim de validar os conceitos propostos. Foram tratados os seguintes casos:

- controle do braço de um *robô* (Apêndice I);
- *jogo da velha*;
- problema da máquina de *K-braços* [13];
- controle de agentes em diversos ambientes.

Para todos estes casos, o agente foi capaz de realizar as tarefas propostas através de um processo contínuo de aprendizagem.

6. Conclusões e Discussões

Neste trabalho foram apresentados e discutidos uma série de novos conceitos que podem romper, sob certos aspectos, os paradigmas de modelagem de sistemas de controle convencionais. Para tal, são propostas estruturas de controle baseadas no conceito de metaesquemas de transição. O modelo proposto é bastante genérico e de fácil programação, tendo sido aplicado com sucesso a alguns problemas de controle que, apesar de simples, são bastante representativos.

Uma comparação direta dessa proposta com outras abordagens baseadas em treinamento condicionado é dificultada pelo fato de que a nova metodologia permite um particionamento do problema alvo (com a definição de uma hierarquia de objetivos) que é fortemente dependente da aplicação, não apresentando um equivalente direto com as abordagens tradicionais.

O processo de aprendizagem dos agentes assemelha-se muito ao observado em seres humanos, o que, de certa forma, era esperado, já que a base teórica utilizada está fortemente apoiada nos conceitos propostos por Piaget [8].

As principais dificuldades para a utilização da presente proposta são, em primeiro lugar, a necessidade do usuário romper alguns paradigmas, entendendo claramente a metodologia apresentada e, em segundo lugar, a necessidade da criação de máquinas de inferência que processem metaesquemas, permitindo a implementação eficiente da aplicação.

Referências

[1] C. J. C. H. Watkins, P. Dayan. Q-learning. Machine Learning, 8 (3): 279-292, 1992.

[2] A. G Barto, R. S. Sutton, C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Transactions on Systems, Man, and Cybernetics, SMC-13 (5): 834-846, 1983.

[3] P. Dayan. The convergence of TD(λ) for general λ. Machine Learning, 8 (3): 341-362, 1992.

[4] R. S. Sutton. Temporal Credit Assignment in Reinforcement Learning. Ph.D. thesis, University of Massachusetts, Amherst, MA., 1984.

[5] D. H. Ackley, M. L. Littman. Generalization and scaling in reinforcement learning. In Touretzky, D. S. (Ed.), Advances in Neural Information Processing Systems 2, pages 550-557, San Mateo, Morgan Kaufmann, 1990.

[6] L. P. Kaelbling, M. L. Littman, A. W. Moore. Reinforcement learning: a survey. Journal of Artificial Intelligence Research, 4 (5): 237-285, 1996.

[7] M. Hauskrecht, N. Meuleau, C. Boutilier, L. P. Kaelbling, T. Dean. Hierarchical solution of Markov decision processes using macro-actions. Proceedings of the Fourteenth International Conference on Uncertainty In Artificial Intelligence, 1998.

[8] J. Piaget, B. Inhelder. Memory and Intelligence. Routledge & Kegan Paul, London, 1973.

[9] C. Balkenius. Neural mechanisms for self-organization of emergent schemata, dynamic schema processing, and semantic constraint satisfaction. Lund University Cognitive Studies, 14,. 1992.

[10] R. Sun. On schemas, logics, and neural assemblies. Applied Intelligence, 5 (2): 83-102, 1995.

[11] G. L. Drescher. Made-up Minds - A Construtivist Approach to Artificial Intelligence. The MIT Press, Massachusetts, 1991.

[12] R. S. Wazlawick. Um Modelo Operatório para Construção de Conhecimento. Tese de doutorado, Pós-Graduação em Engenharia de Produção, UFSC. Florianópolis, SC, 1993.

[13] D. A. Berry, B. Fristedt. Bandit Problems: Sequential Allocation of Experiments. Chapman and Hall, London, 1985.

Apêndice I – Exemplos de Aplicação

Consideremos um agente autônomo que se movimenta em um ambiente que contém objetos. O agente possui um simples sistema sensorial (rudimentos de visão, tato e paladar) e um simples sistema motor, realizando ações de deslocamento, controle de braço/mão e boca.

O agente possui as seguintes entradas:

E_1 - estado da mão (1=fechada);

E_2 e E_3 - posição da mão (01=em baixo, 10=em cima);

E_4 - sensor de tato (1=segurando objeto);

E_5 - estado da boca (1=fechada);

E_6 e E_7 - sensor de paladar (01=doce, 10=amargo);

E_8 - visão (1=objeto visualizado);

E_9 e E_{10} - sensores de reforço (01=prazer,10=dor).

O agente possui as seguintes saídas:

S_1 - abre a mão;

S_2 - fecha a mão;

S_3 - sobe a mão;

S_4 - desce a mão;

S_5 - abre a boca;

S_6 - fecha a boca;

S_7 - dar um passo.

Nesse exemplo, foi utilizado um mecanismo de exploração externo ao agente, sendo que o sinal de reforço foi usado para definir o mais alto nível da hierarquia de objetivos. Dessa forma, toda a ação do agente é orientada para a obtenção de um estímulo de *prazer*.

O agente foi programado através de metaesquemas, sendo utilizada a seguinte sintaxe:

(*espaço de causa* | *espaço observado* | *saídas objetivo* | *entradas objetivo*).

A seguir é apresentada a listagem do *programa* de metaesquemas utilizado para controle do agente:

Abre e fecha a mão => estado da mão:
(*S1, S2* | *E1* | *S1, S2* | *O1*)
Sobe e desce a mão => posição da mão:
(*S3, S4* | *E2, E3* | *S3, S4* | *O2, O3*)
Abre e fecha a boca, posição da mão => estado da boca:
(*S5, S6, E2, E3* | *E5* | *S5, S6 , O2, O3* | *O5*)
Anda um passo, posição da mão => vê objeto:
(*S7, E2, E3* | *E8* | *S7, O2, O3* | *O8*)
Estado da mão, posição da mão, vê objeto, gosto do objeto => tato:
(*E1, E2, E3, E8, E6, E7* | *E4* | *O1, O2, O3, O8* | *O4*)
Posição da mão, tato, estado da boca => gosto:
(*E2, E3, E4, E5* | *E6, E7* | *O2 ,O3, O4, O5* | *O6, O7*)
Estado da boca, gosto do objeto, posição da mão => prazer , dor:
(*E5, E6, E7, E2, E3* | *E9 , E10* | *O5, O6, O7, O2, O3* | *1, 0*)

Apêndice II – Pseudo Códigos

Rotina de manutenção de esquemas:
repetir

```
  cf = c
  Se cf <> ci então
     ei = e
     repetir
        ef = e
        t = t+1
     até ef <> ei ou t> Tmax
     Armazenar esquema (ci, cf, ei, ef ,t)
  senão
    ci = cf
  fim se
Até fim do programa
```

Rotina de definição de saídas objetivo:

```
Repetir
  Se (Ee<>E) e (Ee<>valor invalido) então
    Início
      Valor Objetivo [Ee] = 1
      para cada esquema Et fazer
        Se Valor Objetivo [cf] >0 então
          Valor Objetivo [ei]=Valor Objetivo[ef ] * p

     Se Valor Objetivo [e] >0 então
       Para cada esquema Et fazer
          Se Ee = e então
              Incluir Et numa lista de sorteio
          repetir
        Sortear um Et da lista
        repetir
           Se c <> ci  então
             Sc = ci
          senão
             Sc = cf
           t=t+
    até (e = ef para o Et sorteado) ou
          (t > Tmax)
     até (e = ef ) ou (e= Ee)
  senão
     Sc = valor invalido
 fim se
Até fim do programa
```