RobotShopping

Edna de Carvalho Andrade <ednacarvalhosempre@gmail.com>
Kaic de Oliveira Barros < kaicbarros@gmail.com>
Marcos Neto Santos <marcos_nto@hotmail.com>

Universidade Federal de Sergipe (UFS) - Curso de Sistemas de Informação – Campus Itabaiana Av. Vereador Olímpio Grande, S/N – Bairro Centro – CEP 495000-000 – Itabaiana - SE

RESUMO

Este documento tem por finalidade apresentar a aplicação de aprendizado por reforço em um simulador de vigia noturno. Para aplicação dessa estratégia foi utilizado o algoritmo de Q-learning para aprendizagem do ambiente proposto. A ferramenta utiliza o método para aprender o caminho de todos os pontos para todos os pontos que foram retirados do ambiente de aprendizagem. Para desenvolvimento e apresentação da ferramenta foi utilizada a planta Shopping Peixoto, da cidade de Itabaina-SE.

Palavras chaves: aprendizado por reforço, algoritmo de Q-learning.

ABSTRACT

Title: *RobotShopping*

This document aims to present the application of learning by reinforcement in a night watch simulator. In order to apply this strategy, the Q-learning algorithm for learning the proposed environment was used. The tool uses the method to learn the path of all points for all points that have been taken from the learning environment. For the development and presentation of the tool was used the plant Shopping Peixoto, from the city of Itabaina-SE.

1 INTRODUÇÃO

A disciplina Inteligência Artificial nos apresenta conceitos e técnicas para que possamos aplicá-los na resolução de problemas computacionais, levando em consideração as vantagens e desvantagens de cada técnica. Um dos conceitos abordados é o do aprendizado por reforço um dos paradigmas computacionais de aprendizagem, estudado pela ciência da computação onde o método de programação consiste em um agente aprender como se comportar num ambiente dinâmico através de interações do tipo "tentativa e erro". Tal conceito foi utilizado para o desenvolver um simulador para melhor abstração do método.

Para o desenvolvimento e apresentação foi utilizada a planta do Shopping Peixoto. Para tanto, será apresentado uma breve explicação do método de aprendizagem por reforço e seu funcionamento, além dos procedimentos para desenvolvimento do simulador, expondo partes de códigos e imagens.

2 APRENDIZADO POR REFORÇO

O aprendizado por reforço é um paradigma computacional de aprendizagem em que um agente aprendiz procura maximizar uma medida de desempenho baseada nos reforços que recebe ao interagir com um ambiente desconhecido. O agente tem como objetivo aprender de maneira autônoma uma política ótima de atuação, através da interação com o ambiente.

No ambiente de aprendizado por reforço, um agente é inserido em um ambiente e interage com ele através de percepções e ações. A cada passo o agente recebe como entrada uma indicação do estado atual do ambiente. O agente então escolhe uma ação a tomar, e gera a sua saída. A ação altera o estado do ambiente, e uma medida dessa mudança de estado é informada ao agente através de um valor de sinal de reforço. O comportamento do agente deve tomar ações que maximizem o valor final da soma dos

reforços recebidos em um intervalo de tempo ou tentativas. Tal politica deve ser aprendida através de um processo de tentativa e erro.

Formalmente, o aprendizado por reforço pode ser identificado alguns elementos como por exemplo:

- Uma política responsável por definir o padrão de comportamento do agente, ou seja, uma política π determina como o agente deve decidir por certas ações, em detrimento de outras.
- Uma função valor que associa um valor a um estado (par estado-ação).
- O modelo do ambiente no qual o agente aprendiz será inserido.

A aprendizagem no aprendizado por reforço se dá sem a presença de um professor que ensina através de exemplos. Na resolução de um problema de aprendizagem por reforço a meta maior é levar o agente a escolher a sequência que tendem a aumentar a soma de valores de reforço, ou seja, encontrar uma política ótima, π* que maximize os sinais de reforço acumulados ao longo do tempo. Sua utilização é recomendada quando não se dispõe de modelos a priori, ou quando não se consegue obter exemplos apropriados das situações as quais o agente aprendiz irá enfrenta.

2. 1 ALGORITMO DE Q-LEARNING

O algoritmo Q-learning desenvolvido por Watkins é considerado uma das mais importantes contribuições em aprendizado por reforço. Ele é um dos algoritmos de aprendizagem por reforço que não necessitam de uma modelagem completa do ambiente, ou seja, não necessita conhecer a matriz de probabilidades de transição e r (s, a) para todos os possíveis estados e ações do ambiente.

O Q-learning para valores ótimos de Q não depende da política que está sendo utilizada, a função ação-valor Q se aproxima diretamente a função ação-valor ótima, através de atualizações dos pares estado-ação, que são feitas à medida que estes pares são visitados. Assim, aprender a função ação-valor Q corresponde a aprender a política ideal (π*). A expressão de atualização do Q-valores no algoritmo de Q-learning fundamenta-se na função ação-valor e é denotada por:

$$\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$$

Uma característica importante deste algoritmo é que a escolha das ações a serem executadas durante o processo de aproximação interativa da função Q pode ser feita através de qualquer critério de exploração/explotação, inclusive de forma aleatória. Isso causa um dilema de quando deve-se explorar (escolher a ação randômica) e quando deve-se usufruir (escolher a ação que atualmente está com maior valor).

3 PROJETO

O objetivo do projeto é simular o aprendizado de máquina em um ambiente, depois utilizar a base de dados aprendida para percorrer os caminhos do ambiente, de forma que simule um vigia. Para o desenvolvimento do simulador, utilizamos uma planta de um ambiente (Shopping Peixoto) para aprendizado do mesmo. Nela foram inseridos pontos que representam os locais de acessos possíveis.

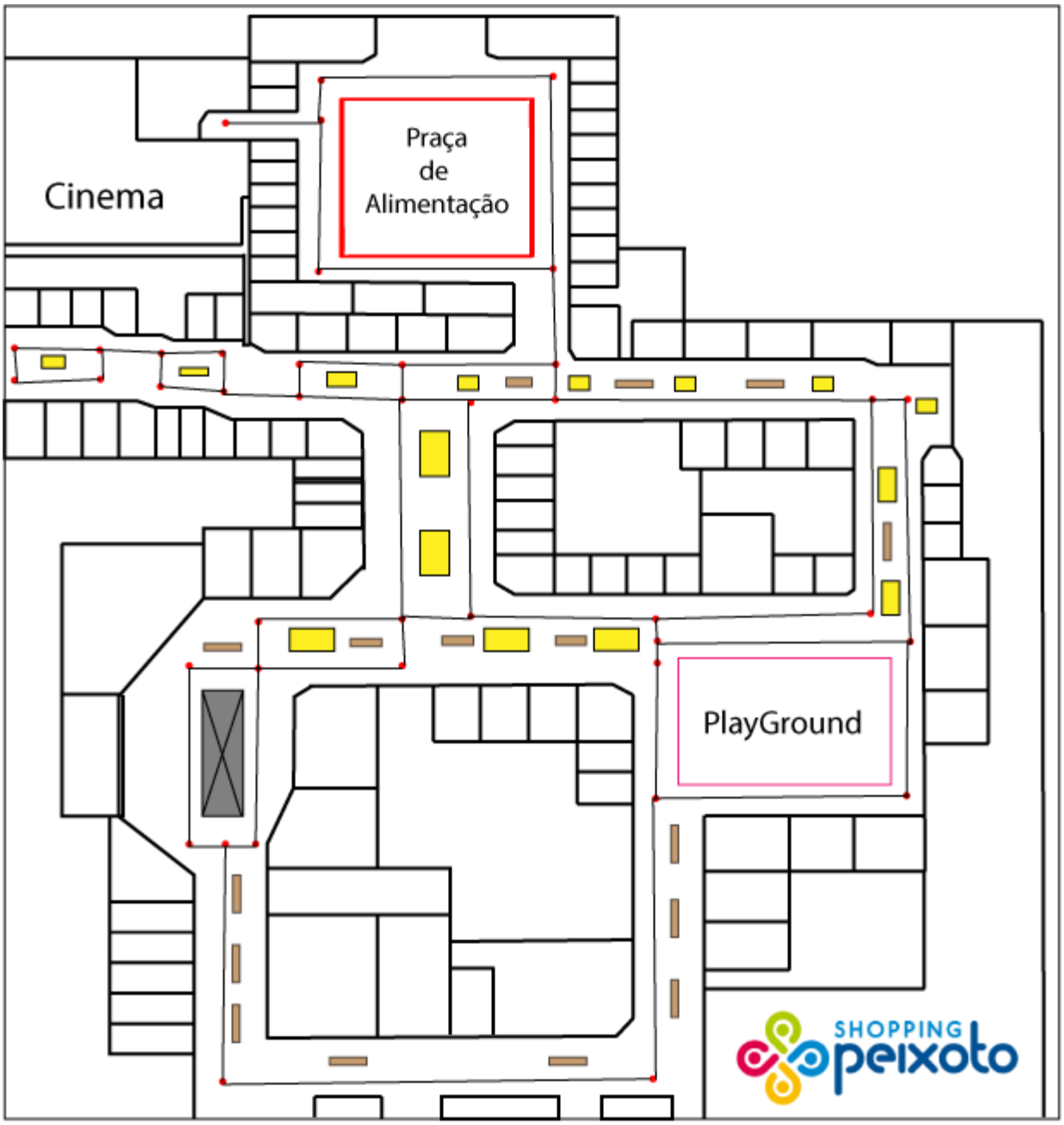

Figura 1 - Planta do Shopping com o grafo

O software foi implementado na linguagem java, utilizando o software Netbeans IDE. Apresenta dentre suas classes a Dado, a Estado e a MetodosPrincipais. As duas primeiras classes são para estanciar um dado, composto do valor da ação e o identificador de um estado, e um estado do ambiente de aprendizado. Já a classe MetodosPrincipais como o próprio nome possui os métodos principais, dentre eles o método aprendizado, buscarCaminho e gerarCaminho.

O método aprendizado possui como parâmetros uma única variável que indica o estado objetivo dentre o conjunto de estados existente. O aprendizado é baseado no algoritmo Q-learning anteriormente apresentado. Ele realiza um laço de repetição, onde o ponto de parada foi definido como chegar 50000 de vezes ao objetivo, dentro dele existe outro laço de repetição que é executado até o estado atual, estado selecionado aleatoriamente na primeira execução, seja igual ao objetivo passado como parâmetro. Destro dele ocorre, a escolha aleatória (exploração), ou especifica (usufruir), de uma ação

dentre as do estado atual. Depois disso, a soma da recompensa do próximo estado, dado a partir da ação escolhida do estado atual que está no momento, somado com o peso (0,9) multiplicado por um valor (o maior valor dentre os valores das ações do próximo estado), é atribuída a ação escolhida do estado atual. Após o estado atual recebe o valor do próximo estado. Após finalização do laço de repetição principal, as informações obtidas pelo processo são salvas em um arquivo especifico do estado objetivo.

O método buscarCaminho realizar o processo de busca de um estado origem, A, até o estado

```
public static void aprendizado(int destino) throws IOException {
    totalUsufruir = 0;
    totalExplorar = 0;
    int x = 0;
    Random seleciona = new Random();
    int estadoAtual;
    while (x < 10000) {
        estadoAtual = seleciona.nextInt(TOTALESTADOS);
        while (estadoAtual != destino) {
            int proximoEstado = -1;
            int acao;
            if (totalAcao(estadoAtual) == 1) {
                acao = retornarAcao(estadoAtual);
                proximoEstado = tabela[estadoAtual][acao].getEstado();
                unicaEscolha++;
            } else {
                int decisao = escolha();
                if (decisao < 30) {
                    acao = maiorValorAcao(estadoAtual);
                    proximoEstado = tabela[estadoAtual][acao].getEstado();
                    totalUsufruir++;
                } else {
                    totalExplorar++;
                    acao = seleciona.nextInt(TOTALACOES);
                    boolean ok = true;
                    while (ok) {
                        Dado d = tabela[estadoAtual][acao];
                        if (d.getEstado() != -1) {
                            ok = false;
                            proximoEstado = d.getEstado();
                        } else {
                            acao = seleciona.nextInt(TOTALACOES);
                        }
                    }
                }
            }
            double recAt = estados[proximoEstado].getRecompensa() +
            (0.9 * (maiorValor(proximoEstado)));
            tabela[estadoAtual][acao].setValorAcao(recAt);
            estadoAtual = proximoEstado;
        }
        x++;
    }
    gravarArquivo(destino);
}
```

destino B. Ele chama o método lerArquivo para ler a tabela de aprendizado correspondente ao estado de destino. E busca através dos valores das ações de cada estado, lidos do arquivo, encontrar o caminho de A até B.

```
public static String buscarCaminho(int origem, int destino) throws IOException {
    lerArquivo(destino);
    String caminho = "";
    while (origem != destino) {
        int estadoAux = -1;
        double maior = -1;
        for (int i = 0; i < TOTALACOES; i++) {
            if (tabela[origem][i].getValorAcao() > maior && tabela[origem][i].getEstado() != -1) {
                maior = tabela[origem][i].getValorAcao();
                estadoAux = tabela[origem][i].getEstado();
            }
        }
        caminho = caminho + "" + estados[origem].getNome() + ";";
        origem = estadoAux;
    }
    return caminho + estados[destino].getNome() + ";";
}
```

O método gerarCaminho realiza o processo de gerar o caminho o qual o rodô deverá percorrer e armazenar em um arquivo. Esse caminho é gerado a partir da escolha aleatoriamente de um estado para ser a origem do processo de vigilância e um estado destino, esses e os demais pertencentes ao conjunto de estados (pontos), armazenados em uma lista, os quais ele deve monitorar. Enquanto ele não passar por todos os pontos da lista, o estado destino se torna o de origem e é escolhido um novo estado para se tornar o destino, sendo que os destinos anteriores não pertencem mais a lista para que não apresente no final locais sem monitoração. O processo é realizado até que a lista de pontos a serem monitorados esteja vazia.

```
public static void gerarCaminho() throws IOException {
    lerEstados();
    ArrayList percorrer = new ArrayList<Integer>();
    for (int i = 1; i < TOTALESTADOS; i++) {
        percorrer.add(i);
    }
    try {
        File f = new File("Caminho.txt");
        FileWriter fw = new FileWriter(f);
        PrintWriter pw = new PrintWriter(fw);
        int ponto = TOTALESTADOS - 1;
        String percurso = "";
        int origem = 0;//sempre iniciar do ponto 1
        ponto--;
        int destino = (int) percorrer.get(ponto);
        ponto--;
        percurso = buscarCaminho(origem, destino);
        while (ponto != -1) {
            String parte[] = percurso.split(";");
            for (int i = 0; i < parte.length; i++) {
                if (i + 1 < parte.length) {
                    pw.print(parte[i] + ";");
                    pw.print(parte[i + 1] + ";");
                    pw.println();
                }
            }
            origem = destino;
            destino = (int) percorrer.get(ponto);
            Collections.shuffle(percorrer); //embaralha
            ponto--;
            percurso = buscarCaminho(origem, destino);
        }
        String parte[] = percurso.split(";");
        for (int i = 0; i < parte.length; i++) {
            if (i + 1 < parte.length) {
                pw.print(parte[i] + ";");
                pw.print(parte[i + 1] + ";");
                pw.println();
            }
        }
        pw.flush();
        pw.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

4 CONCLUSÃO

Este projeto teve como objetivo auxiliar na compreensão das técnicas e conceitos aprendizagem discutidas em sala, sendo o aprendizado por reforço acompanhado do algoritmo de Q-learning utilizados no desenvolvimento da ferramenta. Estes foram utilizados para fornecer dados para a aplicação dado o ambiente de aplicação.

5 REFERÊNCIAS

LIMA JUNIOR, F. C. **Algoritmo Q-learning como Estratégia de Exploração e/ou Explotação para as Metaheurísticas GRASP e Algoritmo Genético**. Universidade Federal do Rio Grande do Norte, Centro de Tecnologia, 2009. Disponível em: < ftp://ftp.ufrn.br/pub/biblioteca/ext/bdtd/FranciscoCLJ.pdf>. Acesso em 15 de set. 2017.

Aprendizagem por Reforço. Disponível em: <http://professor.ufabc.edu.br/~ronaldo.prati/InteligenciaArtificial/reinforcement-learning.pdf.> Acesso em 15 de set. 2017

Aprendizado por reforço. Disponível em: <https://www.maxwell.vrac.puc-rio.br/19637/19637_4.PDF>. Acesso em 15 de set. 2017.