

## Atividade 8: Aprendizado Por Reforço

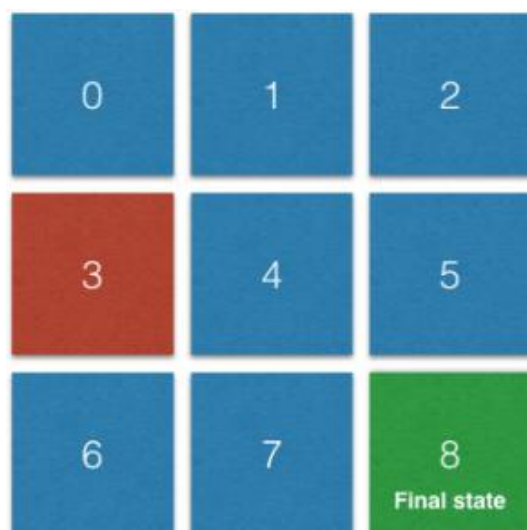
Problemas de aprendizagem por reforço podem ser solucionados pelo algoritmo qlearning. Este trabalho apresenta uma implementação do algoritmo de política de iteração em linguagem Java 1.8, para a disciplina de tópicos especiais em aprendizagem.

### Introdução

Este artigo mostra como a aplicação que implementa aprendizado por reforço por meio da q-learning foi codificada. Para isto, uma breve explicação é feita sobre cada método desenvolvido, que compõe o exercício. O objetivo do trabalho é implementar um modelo simples que atenda os requisitos da disciplina.

### Métodos

Imagine um robô de busca leve que deve navegar de um quarto para outro para parar na sala mais brilhante. Veja abaixo o mapa dos quartos. Podemos considerá-lo um labirinto através do qual nosso robô que busca luz precisará navegar.



Cada sala pode ser considerada

um estado (S), a mudança de uma sala para outra é considerada uma ação (A). Para simplificar, o robô pode mover uma sala por vez, esquerda, direita para cima ou para baixo. O algoritmo precisará de uma matriz de recompensa R e produzirá uma matriz de quantidade Q.

A matriz R contém para cada estado uma linha com as seguintes codificações:

- -1 para as transições impossíveis. Por exemplo, de 0 a 0, temos -1. De 0 a 4, não podemos navegar diretamente, então temos aqui também -1. O mesmo é para a navegação de 0 a 8.
- 0 para possíveis transições. Por exemplo, podemos ir de 0 a 1, então teremos 0 na coluna da matriz 0 coluna 1. Outro exemplo, podemos navegar diretamente de 4 a 7, a linha 4 coluna 7 é 0.
- 100 é a recompensa pelo estado final (a sala mais brilhante). Podemos navegar de 5 a 8 para que a matriz de recompensa tenha na linha 5 coluna 8 o valor 100. A mesma situação para a linha

7 coluna 8.

- -10 é a penalidade (suponha que tivéssemos um quarto com menor intensidade de luz em comparação com os outros quartos). Como você pode ver, podemos chegar à sala 3 das salas 0, 4 e 6. Por essa razão, a coluna 0, linha 3, tem o valor -10 e o mesmo para a linha 4 da coluna 3 e a coluna 6 da coluna 3.

### Desenvolvimento

Os métodos foram implementados na linguagem de programação Java. Diferentes métodos foram criados que em conjunto implementam qlearning.

### CalculaQ

```
void calculateQ() {
    Random rand = new Random();

    for (int i = 0; i < 1000; i++) { // Train
        cycles

        int crtState = rand.nextInt(statesCount);

        while (!isFinalState(crtState)) {
```

```
int[]
actionsFromCurrentState =
possibleActionsFromState(crtState);

int index =
rand.nextInt(actionsFromCurrentState.length);

int nextState =
actionsFromCurrentState[index];

double q =
Q[crtState][nextState];

double maxQ =
maxQ(nextState);

int r =
R[crtState][nextState];

double value = q + alpha *
(r + gamma * maxQ - q);

Q[crtState][nextState] =
value;

crtState = nextState;

}
```

```

    }

}

```

## Policy do estado

```

int getPolicyFromState(int state) {

    int[] actionsFromState =

    possibleActionsFromState(state);

    double maxVal = Double.MIN_VALUE;

    int policyGotoState = state;

    for (int nextState : actionsFromState) {

        double value = Q[state][nextState];

        if (value > maxVal) {

            maxVal = value;

            policyGotoState = nextState;

        }

    }

    return policyGotoState;
}

```

## Resultados

Foi realizado o small grid world fornecido em aula. Esse

small grid world é uma matriz 4x4, onde o primeiro e o último elemento representam o objetivo e o restante são os possíveis estados que o agente pode tomar.
















As ações do problema são:

- Norte
- Sul
- Leste
- Oeste.

Small Grid World utilizado:

|   |   |   |   |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

**Política resultante:**

|   |  |   |   |
|---|--|---|---|
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Conclusão

A partir do algoritmo implementado foi possível executar o exemplo indicado com sucesso, absorvendo os conceitos teóricos e as aplicações. Os resultados de qlearning mostraram a possibilidade de resolver o problema, conseguindo chegar em uma política que solucionou o problema assim como o de política de iteração.

## Referências

1. Scikit. 1.9, Naive Bayes - Disponível em: [http://scikit-learn.org/stable/modules/naive\\_bayes.html](http://scikit-learn.org/stable/modules/naive_bayes.html). Acesso em: 01 de Nov. de 2017
2. Naive Bayes classifier - Disponível em ([https://en.wikipedia.org/wiki/Naive\\_Bayes\\_classifier](https://en.wikipedia.org/wiki/Naive_Bayes_classifier)). Acesso em: 01 de nov. de 2017