Projecto Inteligência Artificial (LEIC 3^o Ano, 1^o Semestre 2017/2018)

Dúvidas Tagus: manuel.lopes@tecnico.ulisboa.pt

Dúvidas Alameda: ernestomorgados@tecnico.ulisboa.pt fausto.almeida@tecnico.ulisboa.pt

November 6, 2017

Abstract

1 Aprendizagem Automática

Neste projecto vamos testar alguns dos algoritmos de aprendizagem discutidos nas aulas teóricas. Iremos testar métodos de classificação (P1), regressão (P2) e aprendizagem por reforço (P3).

1.1 Bibliografia e ambiente de desenvolvimento

A matéria necessária ao desenvolvimento do projecto pode ser encontrada no livro de texto adoptado Artificial Intelligence a Modern Approach. O projecto deverá ser implementado em Python 3.6.2 usando as funções dadas e a toolbox scikit-learn 0.19.0. O desenvolvimento do projecto foi feito usando o sistema anaconda3.

Para a parte de classificação e regressão os métodos estudados nas aulas, e no livro poderão ser utilizados e usando as implementações disponíveis em `http://scikit-learn.org/stable/supervised_learning.html`

2 Entregas e Prazos

Deverá ser submetido 1 zip (com o nome CGGG em que C é A - Alameda ou T - Tagus, e GGG é o número do grupo) com 3 ficheiros contendo o código de cada problema sem alterar o nome dos ficheiros assim como 1 pdf com o relatório . Os ficheiros de código deve conter em comentário, na primeira linha, os números e os nomes dos alunos do grupo, bem como o número do grupo. Não é necessário incluir os ficheiros disponibilizados pelo corpo docente. O relatório em pdf deverá ter o mesmo nome.

As entregas têm que ser feitas até ao limite definido a seguir, data e hora, não sendo aceites projectos fora de prazo sob pretexto algum.

Entrega - até até às 23:59 do dia 09/12/2017

2.1 Condições de realização e discussão dos projectos

Projectos muito semelhantes serão considerados cópia e rejeitados. A detecção de semelhanças entre projectos será realizada utilizando software especializado e caberá exclusivamente ao corpo docente a decisão do que considera ou não cópia. Em caso de cópia, todos os alunos envolvidos terão 0 no projecto, serão reprovados na cadeira e referenciados para o conselho pedagógico.

Os trabalhos serão realizados em grupos de 2 pessoas mas cada pessoa deverá ser capaz de explicar todo o trabalho.

Alguns alunos serão chamados, de forma aleatória ou caso seja necessário confirmar a aquisição a competências, **individualmente** para uma discussão oral do trabalho e/ou uma demonstração do funcionamento do programa.

3 Relatório

O relatório tem um limite máximo de 2 páginas com duas colunas cada.

4 P1 - Métodos de Classificação (5 valores)

Nesta secção vamos estudar métodos de classificação. Tendo um conjunto de dados do tipo:

Entrada	Saída
'rir'	False
'útil'	True
'seara'	False

Ser capaz de predizer se outras palavras, por exemplo 'por' e 'culto' são True ou False. Iremos utilizar a seguinte notação: entrada x, saída y, θ parâmetros da regressão linear a ser aprendidos, f(x) vector de features correspondentes à entrada x.

Vamos utilizar uma regressão linear (com ou sem regularização) num espaço de features.

$$y = sign(\theta^T f(x))$$

Por exemplo se as features forem o número de letras e se o número de vogais é par então é muito fácil treinar um classificador para esta tarefa.

As funções a realizadar deverão ser implementadas em CLASSSOL.PY. O script TESTCLASSSOL.PY testa a solução.

4.1 Escolha de features (1 valores)

Para resolver esta questão é preciso um pouco de trabalho de detective para definir um conjunto de features para descrever as palavras. Por exemplo um vector que conte o tamanho, o tipo de letras e outras coisas parecidas. Essa intuição deverá ir para a função seguinte:

```
def features(X):

    F = np.zeros((len(X),5))
    for x in range(0,len(X)):
        F[x,0] = len(X[x])
        F[x,1] = # implementar
        F[x,2] = # implementar
        F[x,3] = # implementar
        F[x,4] = # implementar
```

4.2 Método de aprendizagem (2 valores)

Poderá ser escolhido qualquer método de aprendizagem e ter-se-á como objectivo encontrar um classificador com mais de 70% de classificações correctas.

```
def mytraining(f,Y,par):
    reg = # implementar
    reg = reg.fit(f, Y)

    return reg

def myprediction(f,reg):
    Ypred = ref.predict(f) # os métodos no scikit learn têm todos uma função predict

    return Ypred
```

4.3 Análise e escolha de parâmetros (2 valor)

A análise deve incluir a comparação dos resultados, uma explicação da forma como parâmetros foram ajustados.

5 P2 - Métodos de Regressão (8 valores)

Nesta secção vamos estudar métodos de regressão do tipo:

$$y = g(x)$$

onde x é o vector de entrada, y é a saída e $g()$ é a função a aprender.
As funções a realizadar deverão ser implementadas em REGSOL.PY. O script TESTREGSOL.PY testa a solução.

5.1 Método de aprendizagem (5 valores)

Deverá utilizar 2 métodos de regressão e fazer um estudo comparativo. Caso os métodos tenham parâmetros ajustáveis deverá fazer a afinação desses parâmetros (e.g. usando validação cruzada).

5.2 Análise e escolha de parâmetros (3 valor)

A análise deve incluir a comparação de resultados, uma explicação da forma como os parâmetros foram ajustados

6 P3 - Aprendizagem por Reforço (7 valores)

Vamos neste exercício aprender por interacção com o mundo. Vamos imaginar que um robot interagiu com o mundo e recebeu (ou não) diferentes recompensas. Essas recompensas estão disponivéis num ficheiro onde cada linha contem o estado inicial, a acção executada, o estado seguinte e a recompensa. Como é uma sequência o estado seguinte numa linha é o inicial na linha seguinte. Pela interacção do agente com o mundo poderiamos perceber a forma do mundo em que ele vive. Temos no entanto que responder as seguintes perguntas.

- Qual é o valor de cada acção em cada estado do mundo?
- Qual é a politica a seguir em cada estado?
- Seguindo a política aprendida para onde é que o agente vai se começar no estado 3?

As funções a realizadar deverão ser implementadas em RLsol.py. O script testRLssol.py testa a solução. O ficheiro RL.py não deverá ser alterado.

6.1 QLearning (2 valor)

Para isso implementar o algoritmo Q-Learning que a partir de uma trajectória recebida calcule os valores Q para cada acção.

```
class myRL:

    def __init__(self, nS, nA, gamma):
        self.nS = nS
        self.nA = nA
        self.gamma = gamma
        self.Q = np.zeros((nS,nA))

    def traces2Q(self, trace):
        # implementar esta funcao
        self.Q = np.zeros((self.nS,self.nA))

        return self.Q
```

6.2 Q2Pol (1 valor)

Para calcular a trajectória é necessário uma política. É necessário implementar a função:

```
def Q2pol(Q, eta=5)
return pol
```

6.3 Gerar trajectória (1 valor)

Pode usar a função dada

```
J,trajlearn = fmdp.runPolicy(4,5,Q2pol(Q))
```

6.4 Interpretação dos resultados (3 valores)

Para cada um dos ambientes faça. Por inspecção das trajectórias, faça uma representação gráfica do ambiente no qual o agente se move (1val). Qual é a função de recompensa? (1val)Como é que o agente se move? (1val)