TÉCNICO LISBOA

Inteligência Artificial

Projeto 2 • Relatório

Grupo 44 • Mariana Mendes, 83521 • Miguel Regouga, 83530

Ano Letivo 2017/2018

P1: Métodos de Classificação

Depois de uma análise ao problema em causa, deparámo-nos com algumas *features,* possíveis de implementar, para descrever as palavras. Nomeadamente, o tamanho/número de letras, o número total de vogais, o número total de consoantes, o maior número de vogais seguidas, o maior número de consoantes seguidas e a soma total do valor *ASCII* de cada letra de uma palavra.

Tamanho da palavra	✓	✓	✓	✓	✓	✓
Número de vogais	✓	✓	✓		✓	✓
Número de consoantes	✓	✓		✓		✓
Número de vogais seguidas	✓			✓	✓	✓
Número de consoantes seguidas	✓	✓	✓	✓	✓	✓
Soma do valor dos caracteres ASCII da palavra	✓	✓	✓	✓	✓	
wordsclass.npy	0.0384615384615	0.0432692307692	0.0528846153846	0.0384615384615	0.0384615384615	0.204326923077
wordsclass2.npy	0.0384615384615	0.0432692307692	0.0528846153846	0.0384615384615	0.0384615384615	0.204326923077

Tabela 1 – Estudo dos diferentes valores de erro com as diversas combinações de features testadas.

Após examinar a Tabela 1, chegámos à conclusão de que as cinco *features* a implementar, de modo a obter o menor valor de erro, poderão ser o tamanho, número total de vogais (ou o número total de consoantes, uma vez que o valor de erro é igual com uma ou com outra), o número de vogais seguidas, o número de consoantes seguidas e a soma total do valor *ASCII* de cada letra da palavra em análise.

Para a escolha do método de aprendizagem a implementar, testámos os diferentes classificadores de modo a obter a maior percentagem de classificações corretas, acima do objetivo apresentado. Para tal, construímos a Tabela 2 com os dados obtidos através do estudo dos referidos testes.

	wordsclass.npy	wordsclass2.npy
K Neighbors Classifier (neighbors = 2, weights = uniform)	0.127403846154	0.117788461538
K Neighbors Classifier (neighbors = 2, weights = distance)	0.0456730769231	0.0456730769231
K Neighbors Classifier (neighbors = 3, weights = uniform)	0.122596153846	0.110576923077
K Neighbors Classifier (neighbors = 3, weights = distance)	0.0384615384615	0.0384615384615
Decision Tree Classifier (min_samples_split = 2)	0.0384615384615	0.0384615384615
Decision Tree Classifier (min_samples_split = 8)	0.0721153846154	0.0769230769231

Tabela 2 — Comparação dos valores obtidos com os diferentes métodos de aprendizagem.

Com base na tabela acima, concluímos que os modelos que apresentam o menor erro são o Modelo dos Vizinhos Mais Próximos (implicando uma métrica de distância) com k = 3, e o *Decision Tree Classifier*, com *min_samples_split* = 2. Desta forma, e uma vez que a procura com o *Decision Tree Classifier* é mais complexa e lenta, optámos pelo primeiro método referido.

P2: Métodos de Regressão

Após testar todos os métodos de aprendizagem, referentes aos métodos de regressão, presentes nos ficheiros *regression.py* e *regressionnll.py*, chegámos à conclusão de que o único modelo que satisfaz as condições impostas pelo script *testregsol.py* é o *Kernel RBF*. O segundo modelo escolhido que se aproxima mais de satisfazer

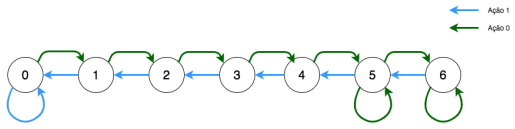
essas mesmas condições é o *Kernel Polynomial*, sendo que este apenas satisfaz uma das condições impostas. Ambos os modelos escolhidos necessitaram de ajustes nos parâmetros *gamma* e *alpha*; desta forma, fomos ajustando estes valores, aumentando/diminuindo-os consoante se os valores dos resultados se aproximavam ou afastavam dos limites impostos.

Alpha	Gamma	Kernel Ridge (RBF)		Kernel Ridge (Polynomial)	
		regress.npy	regress2.npy	regress.npy	regress2.npy
1	1	2.84370545151	2129.14703022	7.74934298062	0.382507737304
0,1	1	1.93747841345	2009.57685977	5.94007071796	0.0767902817662
1	0,1	0.885445510627	1855.76345735	4.32916363248	371.324146071
0,1	0,1	0.232812163582	1264.74000448	5.39564191183	12.5629405803
0,1	0,0003	0.090374124945	448.093181274	0.845678630578	1711.05701148

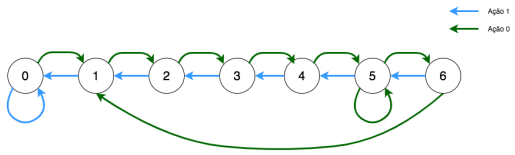
Tabela 3 – Comparação entre o Kernel Ridge (RBF) e o Kerner Ridge (Polynomial).

P3: Aprendizagem por Reforço

Representação Gráfica do Ambiente



Representação gráfica do ambiente 1 descrito com uso dos ficheiros fmdp1.pkl e traj1.pkl



Representação gráfica do ambiente 2 descrito com uso dos ficheiros fmdp2.pkl e traj2.pkl

Função de Recompensa

Depois de uma análise cuidada às tabelas obtidas quando é executado o script *testRLsol.py*, podemos concluir que a função de recompensa é igual para os dois ambientes estudados. Desta forma, chegámos à conclusão que, caso o agente realize uma ação (0 ou 1) no estado 0 ou 6, recebe uma recompensa de valor 1; caso contrário, recebe uma recompensa de valor 0.

Movimento do Agente

Como foi especificado no script *testRLsol.py*, o agente move-se consoante duas ações, 0 ou 1, em ambos os ambientes existentes.

Na ação 1, o agente move-se de igual forma nos dois ambientes referidos, sendo este o seguinte:

- O agente desloca-se "para trás" em direção ao estado 0, por ordem numérica de estados decrescente;
- Quando atinge o estado 0, mantém-se no mesmo estado.

Já na ação 0, o movimento do agente é bastante semelhante nos dois ambientes em causa, diferindo apenas num aspeto. O seguinte comportamento é comum a ambos:

- O agente descola-se "para a frente" em direção ao estado 6, por ordem numérica de estados crescente;
- No estado 5, a ação 0 tem duas opções distintas, ambas com recompensa de valor 0. Estas opções são:
    - Manter-se no estado 5 (nele próprio);
    - Encaminhar-se para o estado 6.

O aspeto em que o deslocamento do agente difere nos dois ambientes é:

- No ambiente 1, ao escolher a ação 0, o agente, chegando ao estado 6 mantém-se no mesmo estado, i.e., no estado 6;
- No ambiente 2, ao atingir o estado 6 e escolher a jogada 0, o agente encaminha-se para o estado 1.