

Universidade de Lisboa - Instituto Superior Técnico  
Licenciatura em Engenharia Informática e de Computadores  
Inteligência Artificial  
2º Projeto - Grupo 22

Gonçalo Marques, 84719

Manuel Sousa, 84740

## P1

Começámos por elaborar um conjunto de features a aplicar sobre as palavras. De início contruímos features básicas que verificassem o número de vogais e consoantes de uma palavra, o número de acentos, etc. O primeiro objetivo passava apenas por estudar o comportamento do avaliador, e durante este processo, facilmente concluímos que quanto mais único fosse o output da feature em relação à palavra recebida, menor seria o erro.

Tabela 1: Análise individual dos erros de cada feature

Features Individuais	Teste 1	Teste 2
[F1] N Acentos	66.6 %	23.1 %
[F2] N Vogais Par	26.4 %	23.1 %
[F3] N Vogais	26.4 %	34.8 %
[F4] N Consoantes	26.4 %	23.1 %
[F5] Palavras Repetidas	26.4 %	23.1 %
[F6] Palavra Par	23.1 %	23.1 %
[F7] Soma ASCII	13.0 %	12.2 %
[F8] Hash	0.0 %	0.0 %

Podemos observar que a função que soma o ASCII dos caracteres constituintes da palavra, tem um erro muito reduzido visto que o output dado pela feature será sempre único, menos quando palavras diferentes são constituídas pelos mesmos caracteres. Assim, uma função que

der um output único para cada palavra recebida iria dar um erro ainda mais baixo. Criámos uma função que gera um inteiro único para uma palavra (Hash), e desta maneira conseguimos obter uma percentagem de erro de 0%.

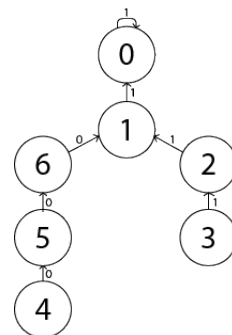
Tabela 2: Análise coletiva dos erros com várias features

Features Coletivas	Teste 1	Teste 2
[F5] + [F6]	23.1 %	23.1 %
[F5] + [F6] + [F7]	7.7 %	7.7 %
[F4] + [F5] + [F7] + [F8]	0.0 %	0.0 %
[F3] + [F4] + [F7]	6.4 %	6.4 %
[F3] + [F4] + [F7] + [F8]	0.0 %	0.0 %

Por observação à tabela concluímos que a utilização de várias features produz um erro mais baixo, que usar features individuais. Observamos também que a feature 8 é predominante, visto que a sua presença é suficiente para dar erro de 0%. De forma a obter os melhores resultados (excluindo a feature 8), fomos ajustando as várias features até obter o melhor resultado possível. Para testar as features utilizámos como classificador o K-Neighbours com 1 vizinho, porque teve um desempenho mais estável ao longo dos nossos testes.

## P2

Nesta secção usámos dois métodos de regressão não-linear: Epsilon-Support Vector Regression (SVR) e KernelRidge com kernel de Radial-basis function (KRR). Usando validação cruzada (K-fold cross validation com um K=13), concluímos que as precisões de cada um destes métodos em ambos os testes são, respetivamente, (-0.17; -62.50) e (-0.12; -37.23). A grande disparidade de precisão entre ambos os testes deve-se ao facto de os dados apresentados estarem dispostos exactamente de acordo com uma função de 3º grau no primeiro teste, ao contrário do 2º teste, em que os valores estão muito mais dispersos. Os parâmetros foram ajustados de maneira a modelarem mais precisamente os dados apresentados em ambas as situações. Concluímos então que o método KRR tem melhor precisão no caso apresentado.



A função de recompensa,  $f(x) = y$  é a seguinte para ambas as trajetórias, em que  $x$  corresponde ao estado em que o agente se encontra e  $y$  é a recompensa de executar uma ação nesse estado.

$$\begin{cases} 1 & x = 0 \vee 6 \\ 0 & x \neq 0 \end{cases}$$

## P3

As imagens seguintes ilustram a maneira como o agente se movimentava pelos ambientes 1 e 2, incluindo uma representação gráfica dos mesmos:

