Relatório para o Projeto de Inteligência Artificial, Parte Dois, Grupo 081

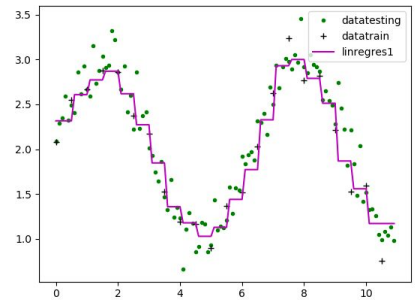
56564, Pedro Filipe Lopes Caetano; 75278, Francisco Braga de Macedo de Castro Henriques

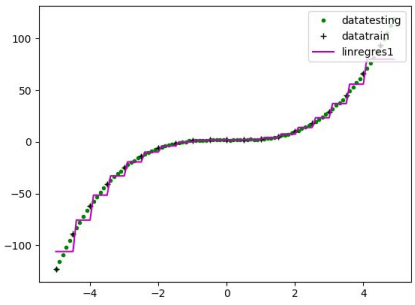
4.3 A study about the characters that composed the binarily labeled words provided for the training data was made. Based on that study we've assumed that, for a given label, some characters wouldn't be allowed to exist in a word or as the first or the last character of that same word, while allowed, respectively, if the opposite label is to be applied. So we decided the set of features to be, along with the number of characters of the word and the predicate of having an even number of vowels as recommended on the assignment document, the following predicates; containing digits, having a first character that only one of the labels and having a last character that only one of the labels would allow.

Regarding the choice of the classifier, sklearn.linear_model.SGDClassifier and sklearn.linear_model.LogisticRegression were tried and yielded the defined to be acceptable error in testclasssol.py while sklearn.naive_bayes.MultinomialNB did not pass the test.

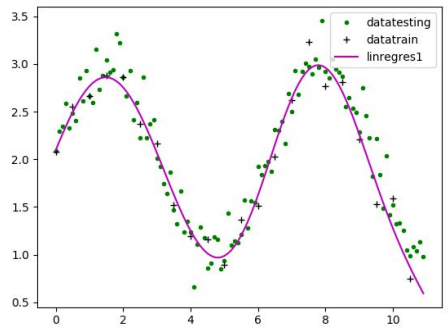
5.2 To be able to learn the function two different methods, Kernel Ridge and K neighbors Regressor, were tested with the objective of getting a good approximation of the provided test data. Kfold Cross validation using Mean Squared error was used to determine a good fit and to adjust any parameters used by the methods.

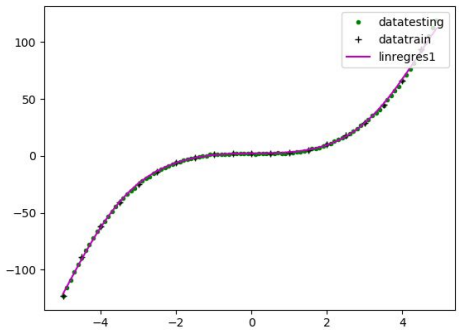
The K neighbors was adjusted with different values for the number of neighbors to take into account, although this was not enough to meet the criteria defined in the testregsol.py file. Below is represented the prediction for the K neighbors for both datasets tested.





The Kernel Ridge model required adjustment of both the alpha and gamma parameters. For this the model was tested using different values for gamma (1,0.1,0.01,0.001) and for alpha (1,0.1,0.01,0.001) with the objective of reducing the score of the cross validation to a minimum. With gamma=0.1 and alpha=0.001 the model passes both tests set in testregsol.py. In the charts below it is possible to see how the predictions fit the datasets tested:

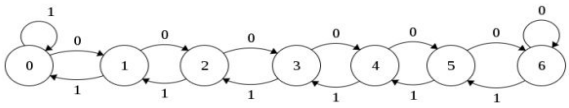




6.1 In RLsol.myRL.traces2Q it was implemented the Q-learning algorithm as it is in RL.finiteMDP.traces2Q except that for the stop condition of its main loop the error threshold was diminished in such a way that the Q-values computed were close enough to those computed by the Value Iteration algorithm (RL.finiteMDP.VI) for the definition of “close enough” present in the file testRLsol.py.

6.2 RLsol.Q2pol was implemented the same way as RL.finiteMDP.Q2pol is, yielding a policy of moving with greater than 98% chance of choosing action ‘0’ if in on of the states {0, 1, 2, 3} or choosing action ‘1’ if in on of the states {4, 5 , 6}.

6.4 Both the environments in which the agent acts are the same and can be depicted with the following finite state machine in which each state is the state the agent find itself in after applying a given action as a transition in the previous state.



The reward function is characterized by the value 1 the agent applies action 0 or 1 from either state 0 or 6. And 0 for all other possibilities.
The environment could interpreted as one dimensional world. So the agent can only try to move left or right (which correspond to the actions 1 and 0), which will happen with success as long as the agent is not trying to cross the frontiers of this world. That is, not applying action 1 when on state 0 and not applying action 0 when on state 6. In these cases the agent hits a wall and stays where it was.