PyGol

An
Explainable Learning Engine
using
Meta Inverse Entailment

Developed By

Dany Varghese & Alireza Tamaddoni-Nezhad

Department of Computer Science
University of Surrey
Guildford, Surrey GU2 7XH, U.K

June 2022

Contact : dany.varghese@surrey.ac.uk

1 Learning Settings

ILP Normal Learning Approach

```
pygol_learn(
                estimator_1, estimator_2, constant_set=[], meta_rule=[],
                min_pos=1, max_neg=0, max_literals=2, exact_literals=False
                , key_size=1, distinct=False, optimize=False, verbose=False,
                eval_fn=”accuracy”, reduce_bc=False, bc_count=20,
                set_chain=False)
               )
```

estimator_1 : It is the estimator returned from the function ”pygol_train_test_split” for the positive training example
estimator_2 : It is the estimator returned from the function ”pygol_train_test_split” for the negative training example

Return the performance estimator of the model.

Example 1.
Test_P, Train_N, Test_N = **pygol_train_test_split** (test_size=0,
positive_file_dictionary=P,
negative_file_dictionary=N)
model= **pygol_learn**(Train_P, Train_N, max_neg=0, max_literals=3, key_size=1)

ILP Learning Approach - Cross Validation

```
pygol_cross_validation(
                estimator_1, file=”BK.pl”, k_fold=10, constant_set=[],
                meta_rule=[], min_pos=1, max_neg=0, max_literals=2,
                exact_literals=False, key_size=[], distinct=False,
                optimize=False, verbose=False, eval_fn=”accuracy”,
                reduce_bc=False, bc_count=20, set_chain=False
                          )
```

estimator_1 : It is the estimator returned from the function ”pygol_folds”

Return the performance estimator of the model.

Example 2.
folds=**pygol_folds**(folds=10)
model=**pygol_cross_validation**(folds, file=”BK.pl”, k_fold=10, min_pos=2,
constant_set=constant_1, set_chain=True, max_literals=2,
distinct=True, optimize=True, max_neg=5)

2 parameters List

1. **file** : **file name** **default=”BK.pl”**

   Background knowledge file name.

2. **container** : **{”dict”, ”memory”}** **default =”dict”**

   Storage type of bottom clause collection to be returned.

3. **positive_example** : **{file, list}** **default=file**

   Positive examples to the model. Either it can be a file or a list. Default is a file with the name ”pos_example.f”.

4. **negative_example** : **{file, list}** **default=file**

   Negative examples to the model. Either it can be a file or a list. Default is a file with the name ”neg_example.f”.

5. **constant_set = list** **default=[]**

   List of constants.

6. **depth** = **integer** **default = 2**

   Number of iterations to be followed while generating the meta clause set.

7. **positive_file_dictionary** = **string** **default = ”positive_bottom_clause”**

   The name of either file or memory variable of positive bottom clause set to be returned. By default, it is ”positive_bottom_clause”. If it is a file, then a pickle file will be generated.

8. **negative_file_dictionary** = **string** **default = ”negative_bottom_clause”**

   The name of either file or memory variable of negative bottom clause set to be returned. By default, it is ”negative_bottom_clause”. If it is a file, then a pickle file will be generated.

9. **tqdm_disable** = **boolean** **default = False**

To control the progress bar. If it is true, progress bar will be hidden.

10. **key** : **integer** **default = 1**

    Builds the bottom clause for positive example number key. Positive examples are numbered from 1, and the numbering corresponds to the order of appearance in the ”positive_example” file.

11. **test_size** : **float** **default = 0.33**

    This value should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split.

12. **shuffle** : **boolean** **default = False**

    Whether to shuffle the data before splitting.

13. **min** : **integer** **default = 1**

    Set a lower bound on the number of positive examples to be covered by an acceptable clause.

14. **max_neg** : **integer** **default = 0**

    Set an upper bound on the number of negative examples allowed to be covered by an acceptable clause.

15. **max_literals** : **integer** **default = 2**

    Sets an upper bound on the number of literals in the **body** of an acceptable clause.

16. **exact_literals** : **boolean** **default = False**

    If it is true, then there will be exactly **N** number of literals in the **body** of an acceptable clause, and N is defined by **max_literals**.

17. **key_size** : **integer** **default = 1**

    Number of bottom clause to be considered to generate meta theory.

18. **distinct** : **boolean** **default = False**

    If it is true, then there will not be any repetitive predicates in an acceptable clause.

19. **optimize** : **Boolean** **default = False**

    If it is true, an optimisation procedure is applied before generating the hypothesis space, this is will speed up the execution.

20. **bc_count** : **integer** **default = 20**

    It will select ”bc_count” number of literals for reducing the size of bottom clause.

21. **reduce_bc** : **boolean** **default = False**

    It will reduce the bottom clause length by selection N number of random literals from meta clause set. N is defined by ”bc_count”.

22. **set_chain** : **boolean** **default = False**

    It will ensure the chaining property of literals in an acceptable clause.