Weekly Update

Khalimat Murtazalieva

November 29, 2020

Contents

1 Task 1

1.1 Description

Perform the same comparison as you have done for AL and non_AL by including the other down-sampling or up-sampling methods you have suggested, so we can see whether AL would also perform better than other sampling technologies.

1.2 Solution

I have chosen three sampling strategies you outlined in README:

- ADASYN
- SMOTE
- CondensedNearestNeighbor

I compared every sampling strategy from the list above with uncertainty sampling from modAL (run 10 fold cross validation and calculated t-test stats for AUC lower boundary, AUC, AUC upper boundary, accuracy, F1, MCC).

All performance metrics for models trained using uncertainty sampling are significantly higher than for models trained using SMOTE. Please see Figure 2 for screenshot with stats (also, see table "t-test stats.csv" in "./results/SMOTE")

All performance metrics for models trained using uncertainty sampling are significantly higher than for models trained using ADASYN except for F1 (0.533 ADASYN vs. 0.566 uncertainty sampling, but difference is not statistically significant). Please see Figure 3 for screenshot with stats (also, see table "t-test stats.csv" in "./results/ADASYN")

All performance metrics for models trained using uncertainty_sampling are significantly higher than for models trained using CondensedNearestNeighbour except for F1 (0.567 CondensedNearestNeighbour vs. 0.565 uncertainty sampling). Please see Figure 4 for screenshot with stats (also, see table "t-test_stats.csv" in "./results/CondensedNearestNeighbour")

2 Task 2

2.1 Description

Include other ML methods and other AL sampling methods

2.2 Solution

I wrote in one of the previous email that uncertainty sampling could work only with ensemble models, of course it is not correct, it works with any probabilistic model, as we just need to find samples nearest to the decision boundary.

3 Task 3

3.1 Description

Try pipeline on the different datasets for SCAMs

3.2 Solution

4 Task 4

4.1 Description

Revisit the scaffold-based train/test splitting to generate more difficult machine learning problems.

4.2 Solution

I decided to implement RDKit butina clustering algorithm[1], since I need scikit-learn 0.23 in environment to run imbalanced-learn, but current deepchem version is incompatible with scikit-learn 0.23 (see. issue here, Bharath kindly informed me that I can use nightly build, but I have already implemented functions

based on butina clustering). The current version of butina split include in the test set only molecules which are "lonely" in their clusters, so we preclude the case when training set include many molecules that are very similar to the ones in the test set (I used function written by Patrick Walters to generate clusters, please, see butina_cluster in utilities.py and wrote a function split_with_butina in main.py). I run 10-fold cross-validation with butina splitter and compared AL- non-AL models performance using paired t-test (see Fig. 1). We observe that model trained using AL strategy perform better based on all metrics. Also, see table "t-test_stats.csv" in "./results/Butina".

References

[1] Darko Butina. Unsupervised data base clustering based on daylight's fingerprint and tanimoto similarity: A fast and automated way to cluster small and large data sets. *Journal of Chemical Information and Computer Sciences*, 39(4):747–750, 1999.

Metrics	Mean non AL	Mean AL	t-test stat	p-value	p_adj	is_significant
AUC_LB	0.697316	0.751998	-5.606194	0.000025	0.000153	True
AUC	0.762249	0.809209	-5.391616	0.000040	0.000241	True
AUC_UB	0.827183	0.866690	-5.092059	0.000076	0.000457	True
Accuracy	0.777007	0.805839	-3.958255	0.000922	0.005530	True
F1	0.418935	0.531619	-5.999662	0.000011	0.000068	True
MCC	0.369016	0.468542	-4.826178	0.000135	0.000813	True

Figure 1: AL and non-AL models performance comparison. Butina splitter

Metrics	Mean non AL	Mean AL	t-test stat	p-value	p_adj	is_significant
AUC_LB	0.706416	0.759994	-5.058256	8.185744e-05	4.911447e-04	True
AUC	0.768604	0.814891	-5.158690	6.593761e-05	3.956256e-04	True
AUC_UB	0.830792	0.870773	-5.289192	4.986293e-05	2.991776e-04	True
Accuracy	0.754909	0.809818	-9.735746	1.346450e-08	8.078699e-08	True
F1	0.531648	0.581098	-3.136904	5.699655e-03	3.419793e-02	True
MCC	0.371471	0.495370	-7.852588	3.190675e-07	1.914405e-06	True

Figure 2: AL and non-AL (SMOTE sampling) models performance comparison. Random splitter

Metrics	Mean non AL	Mean AL	t-test stat	p-value	p_adj	is_significant
AUC_LB	0.704866	0.753524	-4.069654	0.000719	0.004315	True
AUC	0.766866	0.809808	-3.981720	0.000875	0.005248	True
AUC_UB	0.828867	0.866294	-3.843864	0.001189	0.007136	True
Accuracy	0.751636	0.805455	-5.663523	0.000023	0.000136	True
F1	0.533706	0.566469	-1.922907	0.070459	0.422757	False
MCC	0.370061	0.492548	-5.799122	0.000017	0.000102	True

Figure 3: AL and non-AL (ADASYN sampling) models performance comparison. Random splitter

Metrics	Mean non AL	Mean AL	t-test stat	p-value	p_adj	is_significant
AUC_LB	0.719592	0.760051	-4.005001	0.000831	0.004983	True
AUC	0.780296	0.815315	-3.997756	0.000844	0.005064	True
AUC_UB	0.841000	0.870693	-3.942278	0.000955	0.005730	True
Accuracy	0.656364	0.805818	-7.177709	0.000001	0.000007	True
F1	0.567524	0.565532	0.159472	0.875073	5.250436	False
MCC	0.345152	0.497370	-7.042668	0.000001	0.000009	True

Figure 4: AL and non-AL (CondensedNearestNeighbour sampling) models performance comparison. Random splitter