ImbalancedLearningRegression – List of Future Updates

1. List of re-sampling methods from the Python package “imbalanced-learn”.
   Existing methods: CondensedNearestNeighbour, EditedNearestNeighbours, RandomUnderSampler, TomekLinks, RandomOverSampler, SMOTE, ADASYN.
2. Other re-sampling methods and future inventions.
   Existing methods: Introduction of Gaussian Noise, SMOGN (in Nick’s SMOGN).
3. Design a logo for the package.
4. Add a new branch on GitHub after a major release. I suggest using a new branch to write the code and merging it with the master branch by the end of the semester. But for major releases, we can also create a new branch to keep the old one.
5. Add different versions of the documentation. If we modify the codes or implement more methods, we create a separate version of documentation instead of modifying the existing one.
6. Add random states (seeds) for all methods involving randomness. Add a new attribute for a method, and then use that attribute value to control the inner random functions.
7. Object-oriented programming: transform functions to classes. A few suggestions: put all methods in a class, put all over-sampling methods in a class “over-sampler” and all under-sampling methods in a class “under-sampling”, or define a class specifically for a technique and define superclasses. Users will have to create a new instance to call the method. Useful if we are going to implement setters and getters or other functions for a re-sampling method. In “imbalanced-learn”, they adopted classes.
8. The percentage of under-sampling needs to be verified. In SMOGN, the percentage of under-sampling is the data to be deleted. In ImbalancedLearningRegression, the percentage of under-sampling is the percentage of data to keep. For example, if a bin has 100 samples and the percentage of under-sampling for this bin is 60%, then SMOGN will delete 60 samples while ImbalancedLearningRegression will keep 60 samples. Make sure to maintain consistency among all methods in our package. I am 80% sure my solution is correct.
9. Something forked from SMOGN may be an error.

Firstly, at line 165 in “over_sampling_adasyn.py”, rd.random() generate a float number between [0, 1), whereas in the research paper, $\lambda \in [0, 1]$. This problem also exists in “over_sampling_smote.py”.
Secondly, the calculation of a and b on the inverse distance need to be revised. Currently, a and b are only added once no matter how many numeric attributes there are (the loop is meaningless). If we have 100 numeric attributes and 10 nominal attributes, then clearly numeric attributes will dominate the sum of weights. But at line 175 in “over_sampling_adasyn.py”, although we have 100 numeric attributes, the weight is only added once, so the sum of weights will be dominated by nominal attributes. This problem also exists in “over_sampling_smote.py”.
Thirdly, should λ be fixed for all attributes? Currently, it’s fixed for all numeric attributes, but not for nominal attributes. The choice is not fixed at one side for different nominal attributes. See lines 165 and 171 in “over_sampling_adasyn.py”. This problem also exists in “over_sampling_smote.py”. If λ is fixed for all numeric attributes, then we should choose from either the base sample or its neighbor consistently for all nominal attributes, and vice versa. If any change is made here, then we should also review the inverse distance calculation below. This problem also exists in “over_sampling_smote.py”.
Fourthly, for over-sampling a minority bin, SMOGN only adds the synthesized data but does not add the original data. This problem is resolved in ImbalancedLearningRegression.
Fifthly, feat_list_num not only includes all numeric attributes but also includes the target variable. This may be convenient for some code (such as the sixth issue), but for all methods that calculate K-NN using SMOGN’s distance metrics, as well as weighted inverse distance, the target variable should not be included. Please double-check if I am wrong.
Sixthly, suppose feat_list_num excludes the target variable, then look at lines 175 to 179 in “over_sampling_adasyn.py”. If there is no numeric attribute in a dataset, then a and b will not be initialized. UnboundLocalError will be raised below.

10. There is an unknown warning at nominal features' label encoding. This may be due to updates of Pandas. I removed the warning by adding "pd.options.mode.chained_assignment = None" (e.g. line 114 in over_sampling_gn.py). Please check if it is a critical error and update the code accordingly. I didn't see any issue with the new data though.
11. The current implementations of ADASYN and TomekLinks are not efficient because we calculate the pair-wise distances for all samples in our dataset. ADASYN and TomekLinks need to know who the neighbors of a sample are. For a minority sample, we find its neighbors that belong to the majority. SMOTE also needs to know who its neighbors are, but it searches only from the bin the sample belongs to, so it is much faster. On the other hand, CNN and ENN should also be slow, but they used KNeighborsClassifier from scikit-learn, which used special data structures and algorithms to accelerate the process. For SMOTE, ADASYN, and TomekLinks, future collaborators may adopt the KneighborsClassifier or the K-NN classifier from other external packages, allowing them to run faster. They may also reorder or revise the code if it is OK and necessary.
12. ADASYN: compared with the code in imbalanced-learn to see if it is necessary to add more error-checking exceptions like AssertionError, ValueError, and RuntimeError. This is not a big problem and may not be a problem at all!
13. ADASYN: First K-NN search calculates the percentage of neighbors that belongs to the majority. A minority sample not belonging to the current bin occupies a neighbor seat. This could lead to an altered value of r_i. For example, suppose we have one majority bin A and two minority bins B and C, and we are currently over-sampling bin B with K=5. The classes of 7-NN of a sample in B over the entire dataset are: {B, C, C, A, A, A, B}. If minority samples from C are considered, then $r_i = 2/5$. If minority samples from C are ignored, then $r_i = 3/5$. I don't know which one is correct, so please verify it in the future. The second K-NN search is similar to SMOTE, so we only collect minority neighbors belonging to the current minority bin. I am not 100% sure, so please verify it in the future as well.

14. TomekLinks: the under-sampling is not done on an individual majority bin, because there is no bump classification. All minority samples, even if they should belong to different minority bins, are put together and treated as one class. Same for samples that belong to the majority. As a result, we can only find K nearest neighbors of a sample from the entire dataset, not from a majority bin plus all minority bins. A pair of samples may be considered as a T-Link if we only search K nearest neighbors from a subset of the original dataset. But if we search the entire dataset (we include more samples), then the two samples may not be each other's closest neighbor anymore. Moreover, since we may have multiple majority bins and minority bins, how do we then define T-link? Still, two co-neighbor samples belonging to two different big classes: majority and minority? Or belonging to any combination of two bins? Need to be considered again in the future.
    Note that the attribute "option" allows users to choose under-sample either majority or minority samples, which is similar to TomekLinks in "imbalanced-learn".
15. Check the comment in the codes, and add or delete any comment if necessary. "added/modified/exchanged" are written by me to mark the change I made based on SMOGN. I only write them in some of the methods. They can be ignored or removed. Comments having the prefix "SMOGN" are possible errors from SMOGN when I check the code. Comments having the prefix "?????" are my concerns when I check Gloria and Lingyi's code. Some may not be a problem but just my concerns, some of the issues may have already been resolved (some are marked by UPDATE), and some of the issues may still be there. These comments are added by me to make this list. If future experts are so smart that they resolve all issues listed in this file, then all those comments can be removed safely. In addition, there may be comments that do not accord with the contexts, they may be from SMOGN originally and should be deleted. You can also comment on any updates to the code in the future.

Last update: July 1, 2022