Using the imblearn package

Renxiong Liu

The **imblearn** package contains the fast implementation of nonconvex constrained algorithms for both optimizing type I/II error and recall/precision. The goal is to identify the optimal classifiers within a given model class, where the optimality is defined through either type I/II error or recall/precision. The definition of optimal classifier and the details of our proposed method can be found in 1. Liu and Zhu (2022) *On the consistent estimation of optimal Receiver Operating Characteristic (ROC) curve*; 2. Liu and Zhu (2023): *classification for imbalanced dataset*.

The current version only supports implementation of linear model class. The future version will support kernel models and tree-based models.

1 Using imblearn package

1.1 Install imblearn package

To install `imblearn` from github, type in R console

```
devtools::install_github("renxiongliu/imblearn")
```

Note that the installation above requires using R package devtools (which can be installed using `install.packages("devtools")`).

1.2 API: initialize.svm

To use **initialize.svm** function in our **imblearn** package, you shall specify:

1. X: the $n \times p$ feature matrix;
2. Y: the $n \times 1$ binary vector with value -1 and $+1$;
3. w_neg: the class weight for negative label with value in $(0, 1)$;
4. C: the inverse of regularized hyperparameter in SVM.

This will return the slope coefficient `beta` and intercept coefficient `b` for computed initialized linear function.

1.3 API: dccp

To use **dccp** function in our **imblearn** package, you shall specify:

1. X: the $n \times p$ feature matrix;
2. Y: the $n \times 1$ binary vector with value -1 and $+1$;
3. beta_init: slope coefficient for initialized linear function;
4. b_init: intercept coefficient for initialized linear function;
5. gamma: hyperparameter value of regularizer in the object function;
6. psi_k: hyperparameter value of ψ-function $\psi(x, \text{psi\_k}) = \min(1, \max(0, 1 - \text{psi\_k}x))$;

7. max_iter_num: maximal number of iterations in DCCP iterations;
8. max_rel_gap: relative tolerance of object function value improvement during iteration;
9. metrics: the metric to improve for initialized linear classifier. “ROC” means optimizing type I/II error while “PR” means optimize recall/precision.

This will return the slope coefficient `beta` and intercept coefficient `b` for computed improved linear function.

2 Examples of using imblearn package

We briefly illustrate the useage of **imblearn** package with a toy example. To this end, we consider a linear discriminant analysis example:

```
n=100;
p=3;
X=matrix(rep(0,n*p),n,p);
mu=c(-1/(sqrt(p)),1/(sqrt(p)));
Y=rbinom(n,1,0.1);
mu_vec=mu[Y+1];
for(i in 1:n){
X[i,]=c(rnorm(p,mu_vec[i],sd=1));
}
Y=2*Y-1;
```

To initialize a baseline linear classifier, we apply the **initialize.svm** function as follows:

```
w_neg = 0.9
initializer = initialize.svm(X,Y, w_neg=w_neg)
beta_init = initializer$beta
b_init = initializer$b
```

Next, we supply the initializer to our **dccp** function:

```
metrics = "ROC"
solution = dccp(X, Y, beta_init, b_init, metrics = "ROC")
beta_final = solution$beta
b_final = solution$b
```

Our obtained linear classifier determined by beta_final and b_final will now be an optimal classifier in terms of type I/II metrics.