Stage.3.1 More experiments about numerical weighted OCIL algorithm

Timestamp - 2020.11.04

Thanks for the professors suggestion and I try to weighted the numerical attributes to improve the performance of the origin OCIL algorithm. This week I do some more experiments and complete the specific steps in improved parts of the algorithm.Also, I developed the OCIL.py module as the ImprovedOCIL.py module, built the base code of the next experiments.

The improved details about the weighted numerical attributes

To improve the accuracy about the numerical vectors in origin OCIL algorithm, I think we can try to take all features into consideration and weight all attributes properly. By this idea,the features which have larger varience in the same cluster will be valued and play a more essential role in clustering algorithm. Define a weights vector ***w*** like the entropy vector in the origin OCIL algorithm and initially set all weights to $1/\sqrt{N}$. N is the numeber of the attributes. And in iterative steps we can recalculate the new weights vector in ***centroid j*** by $w_{ji} = \exp\left(-h \times X_{ji}\right) / \left(\sum_{l=1}^{N} \left(\exp\left(-h \times 2 \times X_{jl}\right)\right)\right)^{1/2}$. The X_{ji} is defined as follow: $X_{ji} = \frac{1}{|S_j|} \sum_{\mathbf{x} \in S_j} \left(c_{ji} - x_i\right)^2$. S_j is a subset which contains all data samples in centroid j, and the $|S_j|$ is the cardinality of set S_j. The ***h*** is constant which is decided in experiment.

By using the weight vector, we can compare the distance between the samples and centroid in a new way: $L_w(\mathbf{c}_l, \mathbf{x}) = \left(\sum_{i=1}^{N} w_{li}(c_{li} - x_i)^2\right)^{1/2}$. And when we update the cluster subset S_j, we recalculate the weights again, so the results can meet the convergence quickly.

The steps of numerical weighted parts in OCIL algorithm

The categorical part will be the same as the origin OCIL algorithm, and we update the numerical part

Input Dataset as D, k, and h.

1. Start with k initial centroids c1,c2,...,ck;
2. Set the $w_{ji} = 1/\sqrt{N}$, for each centroid c_j,j=1,...,k and each features i=1,...N;
3. for each centroid c_j, find the clusterlabel for each point x by using the origin OCIL similarity metric. But use the new method to calculate the distance:
   $L_w(\mathbf{c}_l, \mathbf{x}) = \left(\sum_{i=1}^{N} w_{li}(c_{li} - x_i)^2\right)^{1/2}$
4. Compute new weights vector, For each centroid c_j, and for each feature i:
   $$\text{Set } X_{ji} = \sum_{\mathbf{x} \in S_j} \left(c_{ji} - x_i\right)^2 / |S_j|$$

$$\text{Set } w_{ji} = \exp\left(-h \times X_{ji}\right) / \left(\sum_{l=1}^{N} \exp(-h \times 2 \times X_{jl})\right)^{1/2}$$

5. refind the clusterlabel of each point X like step 3;
6. compute new centroids by using the same method in origin OCIL algorithm.
7. Iterate 3,4,5,6 until convergence.

The proof of convergence will be cited later.

More experiments and results

Last week I used the former datasets which are use in the origin paper,and I testd some new dataset to get the results for verification now. For Table1 the Statistics information of selected mixed data has been shown. The table2 showed the different errors between original OCIL and the numerical weighted OCIL algorithm from which we can observe that, with proper initializations, the numerical weighted OCIL outperforms the original algorithm in terms of clustering accuracy. Additionally, the datasets which have uneven class distributions in numerical attributes will have better performance in accuracies.

Table1: Statistics of the selected data sets.

Data set	Instance	Attributes(dc+du)	Class	Class Probabilities
Adult	30162	8+6	2	75.11% 24.89%
Blood transfusion	748	1+5	2	24.00% 76.00%
Credit	653	9+6	2	54.67% 45.33%
Statlog	270	7+6	2	55.56% 44.44%
German	1000	13+7	2	70.00% 30.00%

Table2: Clustering errors of original OCIL on mixed data sets in comparison with the numerical weighted OCIL

Data set	original OCIL	Numerical weighted	h value
Adult	0.285061	**0.244049**	12
Blood transfusion	0.326640	**0.281124**	9
Credit	0.245023	**0.231240**	9
Statlog	0.229630	**0.170370**	9
German	0.319000	**0.293000**	22

the raw results and python instance of the experiment

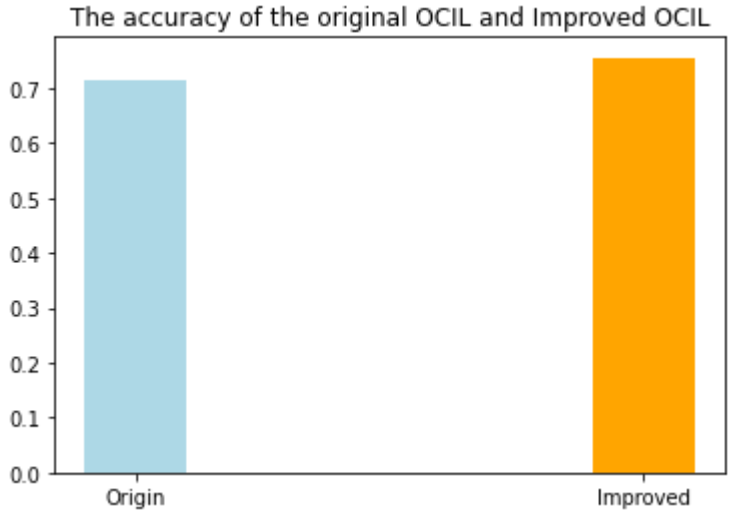
the "adult" dataset

In [1]:
```
import ImpOCIL
import myPersonalTools as Tools
import copy
import numpy as np
import pandas as pd
from sklearn import preprocessing

df_adult = pd.read_table("./adults/adult.data", sep=',')
df_adult = df_adult.dropna()

targetLabel = df_adult["classification"].values
targetLabel = targetLabel.tolist()
dataC = df_adult[["workclass","education","marital-status","occupation","relationship","ra
preDataN = df_adult[["fnlwgt","education-num","capital-gain","capital-loss","hours-per-wee
#for i in range(preDataN.shape[1]):
    #if i == 0:
        #dataN = Tools.myMinMaxScale(preDataN[:,i]).reshape(1,preDataN.shape[0])
    #else:
        #dataN = np.row_stack((dataN,Tools.myMinMaxScale(preDataN[:,i])))
for i in range(preDataN.shape[1]):
    if i == 0:
        dataN = np.array(preprocessing.scale(preDataN[:,i])).reshape(1,preDataN.shape[0])
    else:
        dataN = np.row_stack((dataN,np.array(preprocessing.scale(preDataN[:,i]))))
dataN = dataN.T
clusterC, clusterN = ImpOCIL.clusterUpdate(dataC, dataN, targetLabel)
ImpOCIL.barFigure(dataC, dataN, clusterC, clusterN, 20, targetLabel, 0.1,12, 2)
```

```
The errors of origin algorithm is 0.285061, the improved is 0.244049
```

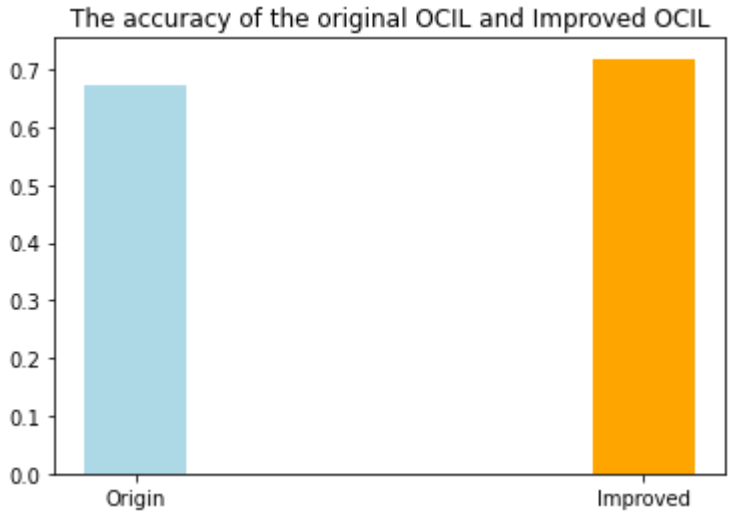


the "Blood Transfusion" dataset

In [4]:
```
import ImpOCIL
import myPersonalTools as Tools
import copy
import numpy as np
import pandas as pd
from sklearn import preprocessing

df_trans = pd.read_table("./transfusion/transfusion.data",sep=",")
preDataN = df_trans[["Recency(months)","Frequency(times)","Monetary(c.c.blood)","Time(mont
dataC = df_trans["Whether he/she donated blood in March 2007"].values
dataC = dataC.reshape(preDataN.shape[0],1)
targetLabel = df_trans["Whether he/she donated blood in March 2007"].values
targetLabel = targetLabel.tolist()
for i in range(preDataN.shape[1]):
    if i == 0:
        dataN = np.array(preprocessing.scale(preDataN[:,i])).reshape(1,preDataN.shape[0])
    else:
        dataN = np.row_stack((dataN,np.array(preprocessing.scale(preDataN[:,i]))))
dataN = dataN.T
clusterC, clusterN = ImpOCIL.clusterUpdate(dataC, dataN, targetLabel)
ImpOCIL.barFigure(dataC, dataN, clusterC, clusterN, 20, targetLabel, 0.05,9, 2)
```

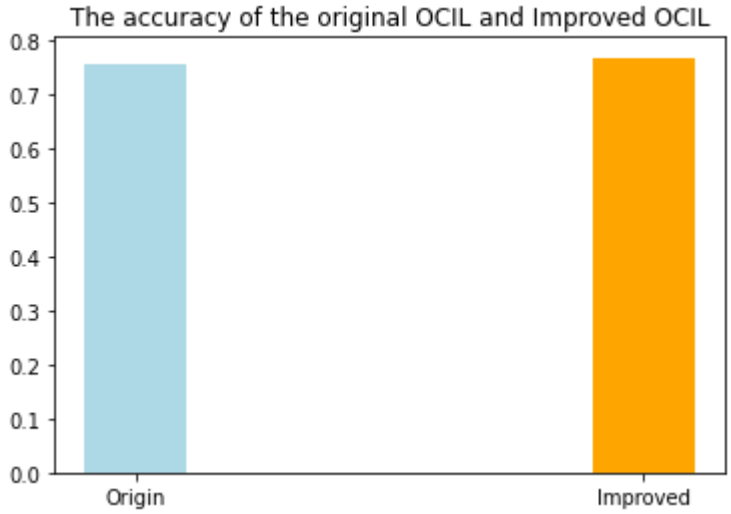
```
The errors of origin algorithm is 0.326640, the improved is 0.281124
```



In [9]:
```
import ImpOCIL
import myPersonalTools as Tools
import copy
import numpy as np
import pandas as pd
from sklearn import preprocessing

df_crx = pd.read_table("./Credit_card_data/crx.data", sep = ",")
df_crx = df_crx.dropna()#delete the rows with missing value
targetLabel = []
for i in range(len(df_crx["A16"].values)):
    if df_crx["A16"].values[i] == "+":
        targetLabel.append(0)
    else:
        targetLabel.append(1)
preDataN = df_crx[["A2","A3","A8","A11","A14","A15"]].values
dataC = df_crx[["A1","A4","A5","A6","A7","A9","A10","A12","A13"]].values
for i in range(preDataN.shape[1]):
    if i == 0:
        dataN = np.array(preprocessing.scale(preDataN[:,i])).reshape(1,preDataN.shape[0])
    else:
        dataN = np.row_stack((dataN,np.array(preprocessing.scale(preDataN[:,i]))))
dataN = dataN.T
clusterC, clusterN = ImpOCIL.clusterUpdate(dataC, dataN, targetLabel)
ImpOCIL.barFigure(dataC, dataN, clusterC, clusterN, 20, targetLabel, 0.6, 9, 2)
```

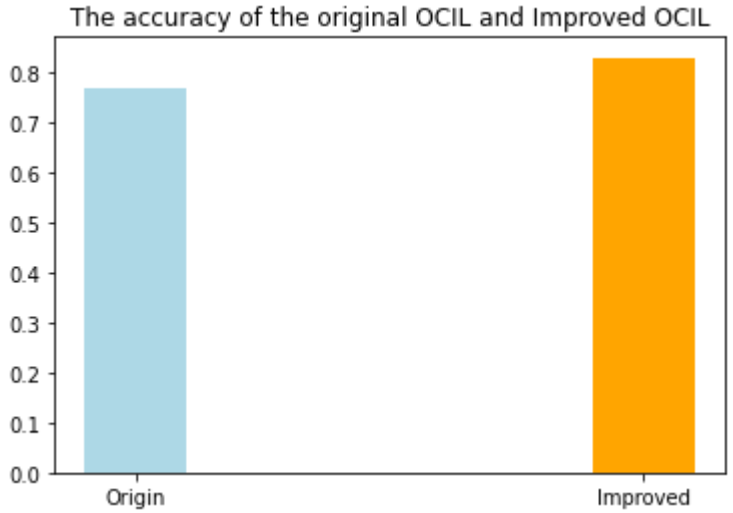
```
The errors of origin algorithm is 0.245023, the improved is 0.231240
```



In [20]:
```
import ImpOCIL
import myPersonalTools as Tools
import copy
import numpy as np
import pandas as pd
from sklearn import preprocessing

df_Statlog = pd.read_table("./Statlog_heart_data/heart.dat", sep = " ")
targetLabel = []
for i in range(len(df_Statlog["Classification"].values)):
    if df_Statlog["Classification"].values[i] == 1:
        targetLabel.append(0)
    else:
        targetLabel.append(1)
preDataN = df_Statlog[["A1","A4","A5","A8","A10","A12","A11"]].values
dataC = df_Statlog[["A2","A3","A6","A7","A9","A13"]].values
for i in range(preDataN.shape[1]):
    if i == 0:
        dataN = np.array(Tools.myMinMaxScale(preDataN[:,i])).reshape(1,preDataN.shape[0])
    else:
        dataN = np.row_stack((dataN,np.array(Tools.myMinMaxScale(preDataN[:,i]))))
dataN = dataN.T
clusterC, clusterN = ImpOCIL.clusterUpdate(dataC, dataN, targetLabel)
ImpOCIL.barFigure(dataC, dataN, clusterC, clusterN, 20, targetLabel, 0.05, 9, 2)
```

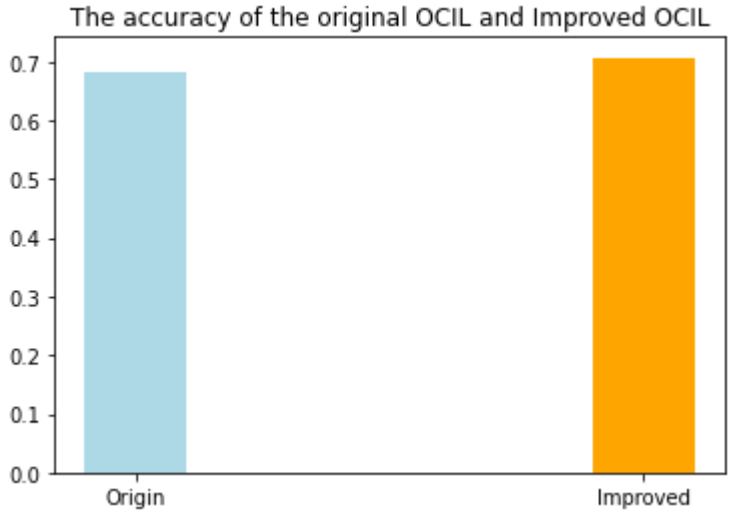
```
The errors of origin algorithm is 0.229630, the improved is 0.170370
```



In [4]:
```
import ImpOCIL
import myPersonalTools as Tools
import copy
import numpy as np
import pandas as pd
from sklearn import preprocessing

df_GerCre = pd.read_table("./German_credit/german.data", sep = " ")
targetLabel = []
for i in range(len(df_GerCre["Classification"].values)):
    if df_GerCre["Classification"].values[i] == 1:
        targetLabel.append(0)
    else:
        targetLabel.append(1)
preDataN = df_GerCre[["Attribute2","Attribute5","Attribute8","Attribute11","Attribute13","
dataC = df_GerCre[["Attribute1","Attribute3","Attribute4","Attribute6","Attribute7","Attri
for i in range(preDataN.shape[1]):
    if i == 0:
        dataN = np.array(preprocessing.scale(preDataN[:,i])).reshape(1,preDataN.shape[0])
    else:
        dataN = np.row_stack((dataN,np.array(preprocessing.scale(preDataN[:,i]))))
dataN = dataN.T
clusterC, clusterN = ImpOCIL.clusterUpdate(dataC, dataN, targetLabel)
ImpOCIL.barFigure(dataC, dataN, clusterC, clusterN, 20, targetLabel, 0.92,22, 2)
```

```
The errors of origin algorithm is 0.319000, the improved is 0.293000
```



Next research plan

- develop the initialization of the original OCIL
- do some research about whether the half-supervised mechanism can be applied

In [ ]: