Ningjia Huang's Individual Report

December 10, 2021

1 Overview

The two algorithms I implemented are Multiple-Instance Learning via Embedded Instance Selection(MILES) and Constructive Clustering-Based Ensemble(CCE). The algorithms were implemented in Python with the use of mil library for loading datasets, numpy for processing arrays, sklearn for building classifiers and split training/testing data, and matplotlib for ploting the result.

2 MILES Algorithm

2.1 Introduction

Multiple-Instance Learning via Embedded Instance Selection (MILES) algorithm tackles the multiple-instance learning problem by embedding bags into an instance-based feature space and selecting the most important features[1]. The algorithm can be divided into 2 stages: feature mapping and feature selection. The feature mapping in MILES extends the Diverse Density(DD) framework. DD assumes there exists a single target concept which can be used to label bags with high accuracy. By maximizing the probability $Pr(t|\mathbf{B_1^+}, \ldots, \mathbf{B_l^-})$, the target that is most likely to match with the data is found. MILES improves on DD by selecting multiple concepts instead of a single concept. The concept class is defined as $C = \{\mathbf{x^k} : k = 1, \ldots, n\}$, where $\mathbf{x^k}$ is the kth instance in the space. The most-likely-cause estimator($Pr(\mathbf{x^k}|\mathbf{B_i})$) between the concept $\mathbf{x^k}$ and the bag $\mathbf{B_i}$ is approximated by:

$$Pr(\mathbf{x}^k|\mathbf{B_i}) \propto s(\mathbf{x}^k, \mathbf{B_i}) = \max_j exp(-\frac{|\mathbf{x_{ij}} - \mathbf{x^k}|^2}{\sigma^2}) \tag{1}$$

, where $\mathbf{x_{ij}}$ is the jth instance in bag $\mathbf{B_i}$. As shown in equation 1, the similarity between a concept and a bag is determined by the instance that is the closest to the concept in the bag. The produced high-dimensional space can be represented by a matrix:

$$[m_1^+, \ldots, m_l^-] = \begin{pmatrix} s(\mathbf{x^1}, \mathbf{B_1^+}) & \cdots & s(\mathbf{x^1}, \mathbf{B_l^-}) \\ s(\mathbf{x^2}, \mathbf{B_1^+}) & \cdots & s(\mathbf{x^2}, \mathbf{B_l^-}) \\ \vdots & \vdots & \vdots \\ s(\mathbf{x^n}, \mathbf{B_1^+}) & \cdots & s(\mathbf{x^n}, \mathbf{B_l^-}) \end{pmatrix} \tag{2}$$

, where each column is the coordinate of a bag $\mathbf{B_i}$ embedded in an instance space. Then, 1-norm SVM is adopted to build classifiers and select features. As in many MIL problems, the training sets are very imbalanced between classes, different penalties are adopted to penalyze on errors produced by positive and negative bags. Therefore, the penalizing factor C is introduced to penalize differently on false negative and false positive.

2.2 Results

The accuracy, precision, recall, AUC scores were tested for different settings of parameters(penalizing coefficient C and scaling factor σ). Table 1 shows the metrics evaluation on different values of C. As the number of positive and negative bags in Fox, Elephant, and Tiger are equivalent, the best performance was given by C around 0.5(that is, penalizing the false positive and false negative equally).

Dataset	Metrics	C = 0.1	C = 0.3	C = 0.5	C = 0.7	C = 0.9
Musk1	Acc.	**0.9 ± 0.089**	0.84 ± 0.12	0.87 ± 0.11	0.86 ± 0.111	0.86 ± 0.092
	Prec.	**0.852 ± 0.168**	0.808 ± 0.152	0.816 ± 0.153	0.867 ± 0.15	0.827 ± 0.124
	Rec.	**0.935 ± 0.1**	0.929 ± 0.088	0.917 ± 0.105	0.89 ± 0.095	0.926 ± 0.092
	AUC	**0.912 ± 0.081**	0.839 ± 0.133	0.878 ± 0.109	0.868 ± 0.115	0.864 ± 0.098
Fox	Acc.	0.56 ± 0.08	**0.57 ± 0.084**	0.475 ± 0.117	0.523 ± 0.083	0.535 ± 0.087
	Prec.	0.594 ± 0.112	**0.556 ± 0.102**	0.471 ± 0.134	0.554 ± 0.103	0.587 ± 0.115
	Rec.	0.558 ± 0.105	**0.579 ± 0.163**	0.445 ± 0.18	0.493 ± 0.029	0.523 ± 0.147
	AUC	0.554 ± 0.085	**0.562 ± 0.085**	0.471 ± 0.122	0.523 ± 0.134	0.531 ± 0.081
Elephant	Acc.	0.79 ± 0.08	0.765 ± 0.071	**0.8 ± 0.063**	0.745 ± 0.085	0.725 ± 0.087
	Prec.	0.748 ± 0.13	0.737 ± 0.118	**0.826 ± 0.117**	0.71 ± 0.138	0.687 ± 0.083
	Rec.	0.886± 0.091	0.737 ± 0.118	**0.799± 0.113**	0.735 ± 0.16	0.743 ± 0.142
	AUC	0.801 ± 0.076	0.824 ± 0.147	**0.804± 0.067**	0.7443 ± 0.09	0.717 ± 0.102
Tiger	Acc.	0.74 ± 0.116	0.75 ± 0.114	0.735 ± 0.071	**0.755 ± 0.076**	0.725 ± 0.068
	Prec.	0.757 ± 0.136	0.773 ± 0.147	0.734 ± 0.128	**0.731 ± 0.127**	0.749 ± 0.108
	Rec.	0.741 ± 0.111	0.741 ± 0.128	0.812 ± 0.146	**0.731 ± 0.107**	0.697 ± 0.063
	AUC	0.743 ± 0.12	0.745 ± 0.112	0.737 ± 0.078	**0.749 ± 0.071**	0.719 ± 0.063

Table 1: Results of MILES with Various Setting of Parameter C

Different values for σ were also tested. For each dataset, C was chosen from Table 1 which gives the best result. The result shows MILES is rather stable with choices of parameter value σ as long as it is chosen from a reasonable range.

Dataset	Metrics	$\sigma = 1$	$\sigma = 3$	$\sigma = 5$	$\sigma = 10000$
Musk1	Acc.	0.85 ± 0.102	0.85 ± 0.136	0.84 ± 0.102	0.39 ± 0.054
	AUC	0.85 ± 0.112	0.826 ± 0.105	0.836 ± 0.101	0.488 ± 0.038
Fox	Acc.	0.545 ± 0.108	0.545 ± 0.091	0.57 ± 0.112	0.41 ± 0.044
	AUC	0.544 ± 0.099	0.552 ± 0.105	0.588 ± 0.098	0.5 ± 0.012
Elephant	Acc.	0.85 ± 0.067	0.805 ± 0.047	0.85 ± 0.05	0.85 ± 0.05
	AUC	0.844 ± 0.082	0.817 ± 0.049	0.847 ± 0.056	0.847 ± 0.056
Tiger	Acc.	0.805 ± 0.072	0.82 ± 0.056	0.8 ± 0.071	0.395 ± 0.057
	AUC	0.811 ± 0.07	0.823 ± 0.061	0.801 ± 0.072	0.5 ± 0.027

Table 2: Results of MILES with Various Setting of Parameter σ

2.3 Extension 1: Consider All Instances in a Bag

One of the limitations of MILES is the most-likely-cause estimator is not well-defined. The similarity function $s(\mathbf{x^k}, \mathbf{B_i})$ only takes the closest instance to a concept into consideration and ignores the other instances in the bag. To take all of the instances in a bag into account, instead of using $\max_j exp(-\frac{|\mathbf{x_{ij}}-\mathbf{x^k}|^2}{\sigma^2})$, we can sum over the distances between the concept and the instances. Since the metrics on Musk1 are relatively satisfying, I decided to combine the max function with a sum function and weigh them differently by introducing two coefficient λ_1 and λ_2. The new most-likely-cause estimator is defined as follows:

$$Pr(\mathbf{x}^k|\mathbf{B_i}) \propto \lambda_1 \max_j exp(-\frac{|\mathbf{x_{ij}} - \mathbf{x^k}|^2}{\sigma^2}) + \lambda_2 \sum_j exp(-\frac{|\mathbf{x_{ij}} - \mathbf{x^k}|^2}{\sigma^2}) \tag{3}$$

By choosing $\lambda_1 = 0.7$ and $\lambda_2 = 0.3$, the result is shown in table 3. The accuracy score on Fox dataset is improved by an average of 0.053 and the AUC score is improved by 0.058; the accuracy score on Tiger dataset is improved by 0.059 and the AUC score is improved by 0.063, which is quite significant.

Dataset	Metrics	C = 0.1	C = 0.3	C = 0.5	C = 0.7	C = 0.9
Fox	Acc.	0.58 ± 0.081	0.6 ± 0.074	0.595 ± 0.085	0.565 ± 0.072	0.586 ± 0.116
	AUC	0.592 ± 0.071	0.604 ± 0.071	0.602 ± 0.088	0.528 ± 0.066	0.607 ± 0.133
Tiger	Acc.	0.81 ± 0.044	0.79 ± 0.086	0.805 ± 0.082	0.815 ± 0.084	0.78 ± 0.078
	AUC	0.822 ± 0.05	0.789 ± 0.09	0.803 ± 0.091	0.816 ± 0.085	0.78 ± 0.081

Table 3: Results with the Combination of Max and Sum Likelihood Estimator

2.4 Extension 2: Ensemble

To further improve the metrics of MILES algorithm, I use ensemble to combine the prediction results from different classifiers and use majority vote to determine the prediction. The three classifiers are 1-norm SVM, C4.5, and Logistic Regression as according to the experiment conducted by James Foulds, these three classifiers outperformed other classifiers on the 4 datasets we chose[2]. Table 4 shows the result after ensemble:

	Musk1	Fox	Elephant	Tiger
Acc.	0.86 ± 0.066	0.595 ± 0.06	0.865 ± 0.088	0.795 ± 0.09
Prec.	0.816 ± 0.149	0.599 ± 0.126	0.872 ± 0.09	0.766 ± 0.138
Rec.	0.95 ± 0.107	0.622 ± 0.161	0.873 ± 0.11	0.825 ± 0.091
AUC	0.861 ± 0.067	0.574 ± 0.074	0.864 ± 0.089	0.774 ± 0.088

Table 4: Results of MILES with Ensemble

Generally speaking, the metrics were improved after ensemble(except for Tiger dataset) but the improvements were not significant. This might be because the classifications made by the three classifiers are similar(as the differences in metrics for the individual classifiers are lower than 1%).

2.5 Conclusion

MILES has relatively satisfying performance on Musk1 but unsatisfying performance on Fox, Tiger, and Elephant(task of image classification). While the algorithm is sensitive to the penalizing factor C, the scaling factor σ does not affect the performance as long as it is chosen from a reasonable range. Extension 1 brings more instances into the decision making process instead of only taking the closest instance(to the target concept) into consideration, which improves the accuracy and AUC scores by 5% to 6%. Extension 2 combines the predictions made by different classifiers where each individual classifier has an overall good performance compared to other classifiers. The improvement is not significant, probably due to the similar predictions made by each of the classifiers.

3 CCE Algorithm

3.1 Introduction

In the Constructive Clustering-Based Ensemble(CCE), the instances in all bags are first collected together and clustered into a self-defined number of clusters(assume to be d)[3]. As clustering may provide the inherent structure of a dataset, the clustered groups may be able to encode information on the distribution of the instances of different bags. The bags are re-represented by a binary feature vector with length of d where each feature is a cluster and is assigned to be 1 if the bag contains instances from that cluster and is assigned to be 0 if not. SVM is used to classify the new binary features.

3.2 Result

Figure 1 shows the result metrics of CCE with number of clusters d ranging from 2 to 80. The plot does not demonstrate an obvious trend, e.g. what range of d results in a better performance compared with others and the variance of the scores is relatively large. A probable interpretation for the low metrics is the performance of CCE highly depends on the clustering results. Since the dimensionality of the instance features is high, k-means may not work well under this condition. As the initial centroids for k-means is randomly chosen, the resulting clusters could be very different, which influences the construction of binary feature vectors as well as the classification. Without 10-fold cross validation, the best accuracy for Musk1 is 0.841(when $d = 43$) and the best accuracy for Musk2 is 0.809(when $d = 44$ and $d = 47$), which are similar to the results concluded from the original paper by Zhou et al.

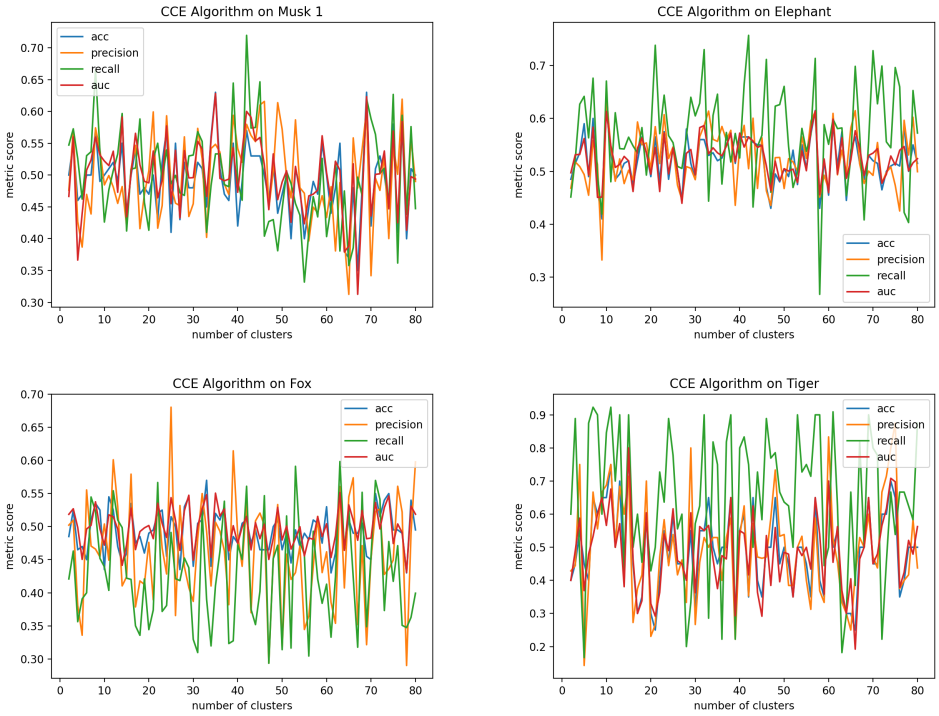

Figure 1: Results of CCE on 4 Datasets

The best results obtained from figure 1 is shown in the following table:

	Musk1	Fox	Elephant	Tiger
Acc.	0.63 ± 0.164	0.555 ± 0.093	0.62 ± 0.171	0.8 ± 0.144
Prec.	0.603 ± 0.156	0.561 ± 0.129	0.643 ± 0.127	0.714 ± 0.119
Rec.	0.62 ± 0.064	0.498 ± 0.133	0.671 ± 0.09	0.89 ± 0.101
AUC	0.623 ± 0.047	0.551 ± 0.062	0.613 ± 0.067	0.8 ± 0.059

Table 5: Best Metrics on Four Datasets By Using CCE

The metrics obtained from CCE is lower(especially for Musk1) compared to other algorithms. The variances of most of the scores are high as the result of randomized initial centroids. It is noticeable that Tiger's best result is close to the results obtained from other algorithms, but a preferrable explanation is that is a coincidence since the average performance on Tiger is still very low.

3.3 Extension 1: kPCA for Feature Selection

As the unsatisfying metrics of CCE could be caused by the poor performance of k-means clustering due to the high dimensional feature vectors, reducing the dimensionality of the original features may help with the clustering process. I used kPCA for the dimensionality reduction. The kernel is defined as a combination of polynomial kernel and a RBF kernel because they capture the different characteristics

of the data: the former demonstrates a better extrapolation ability while the latter demonstrates a better interpolation ability. The new kernel is defined as follows:

$$K_{new} = \lambda K_{poly} + (1 - \lambda) K_{RBF}$$

, where λ is used to control the balance of these two functions. To test the performance of the kPCA-CCE, I evaluated the revised algorithm on Musk1 and Elephant datasets. λ was chosen to be 0.6. The result is shown as follows:

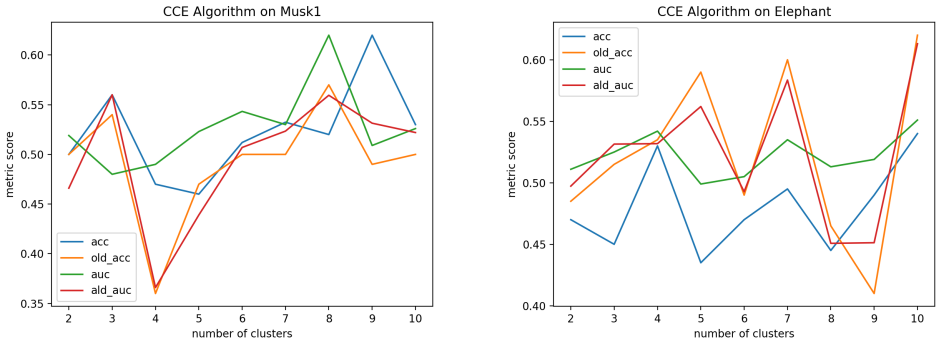

Figure 2: Results of Extension 1 on Musk1 and Elephant Datasets

The accuracy and AUC scores for Musk1 were slightly improved. The accuracy and AUC scores for Elephant did not demonstrate a significant improvement. Therefore, reducing the dimensionality of input features may help with the classification task, but the improvement is not significant.

3.4 Conclusion

The CCE algorithm is inefficient on the MIL task since the rerepresentation of bag feature highly depends on the k-means clustering result. Since the dimensionality of the features of instances is high, the clustering process could be inefficient. By adopting kPCA to reduce the dimensionality of data, the performance on Musk1 was improved and the performance on Elephant remained the same. A future extension that might be able to improve the metrics is to concatenate the newly represented features from different number of clusters since the various clustering results might be able to capture different information from the dataset.

References

[1] Yixin Chen, Jinbo Bi, and J.Z. Wang. Miles: Multiple-instance learning via embedded instance selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):1931–1947, 2006.

[2] James Foulds. Learning instance weights in multi-instance learning, 2007.

[3] Zhi-Hua Zhou and Min-Ling Zhang. Solving multi-instance problems with classifier ensemble based on constructive clustering. *Knowl. Inf. Syst.*, 11:155–170, 02 2007.