Práctica 2: Clasificación multi-instancia

Clasificación no convencional(CNC)

Máster Universitario en Inteligencia Computacional e Internet de las Cosas

Universidad de Córdoba, EPSC

2022/2023

UNIVERSIDAD
D
CÓRDOBA

Autor:

Antonio Gómez Giménez (i72gogia@uco.es)

Índice:

1. Introducción:

El objetivo de dicha práctica es introducir los conceptos de clasificación multi-instancia, para ello, se seleccionaron tres algoritmos de los disponibles en las bibliotecas indicadas, y tres dataset distintos sobre problemas de multi-instancia de los repositorios indicados.

Para la realización de dicha práctica, se pretendió realizar una comparativa entre distintos tipos de dataset diferentes para ver el funcionamiento de los mismos frente a los distintos tipos de algoritmos de aprendizaje basado en multi-instancia.

Respecto a los clasificadores que se pretenden utilizar, estos son:

- **MissSVM** -> se basa en un enfoque de aprendizaje semi-supervisado, tratando las instancias en bags positivas como datos no etiquetados.
- **sbMIL** -> se basa en el sesgo de SVM asumiendo que hay muy pocas instancias positivas por cada bag. En este caso, el conocimiento previo sobre la "escasez" de bolsas positivas puede ser especificado o encontrado mediante validación cruzada.
- **SIL** -> Single-Instance Learning (SIL) se basa en una ingeniosa aproximación de cada etiqueta de cada instancia a la mochila, creando un problema de aprendizaje supervisado pero etiquetando erróneamente instancias negativas en bags positivas. Sorprendentemente suele funcionar correctamente en la mayoría de problemas de este tipo.

2. Experimentos:

Para poder realizar dichos experimentos, se dividió en tres partes, donde cada parte es un dataset distinto:

2.1. Primera parte:

El dataset utilizado en este caso es el dataset **CorelDogs** con 7947 instancias, repartiéndose dichas instancias en 2000 bags distintas. El fichero consta de 11 columnas, siendo la primera la clase de cada patrón, y la segunda, la mochila a la que pertenece. Las 9 columnas restantes hacen referencia a los distintos atributos de cada patrón.

Una vez escogido el dataset, este se carga en el programa, aplicando el procesamiento adecuado para dividir los datos en mochilas con sus respectivas etiquetas. Cabe destacar que este paso es necesario, ya que, los clasificadores de la librería que se pretende utilizar, necesita que se le pase al clasificador, una lista de lista donde cada lista es una mochila con sus respectivas instancias, y una lista que contiene +1 o -1 para definir la clase de dicha mochila (si alguna instancia de una mochila es positiva, entonces dicha mochila es +1).

El código realizado para llevar a cabo lo explicado anteriormente es el siguiente:

```
11      #leer con pandas, junta en listas cada id de bag, transformar a numpy (bags y labels se juntan necesitan numpy)
12      import pandas as pd
13      #dogs = pd.read_csv("CorelDogs.csv", names=["Class", "BagId", "a1", "a2","a3","a4","a5","a6","a7","a8","a9"])
14      #dogs = pd.read_csv("CorelFlowers.csv", names=["Class", "BagId", "a1", "a2","a3","a4","a5","a6","a7","a8","a9"])
15      dogs = pd.read_csv("CorelFood.csv", names=["Class", "BagId", "a1", "a2","a3","a4","a5","a6","a7","a8","a9"])
16      bags = []
17      #total_bags = 0;
18      labels = []
19
20      #divido en labels y datos para cada mochila
21      result = dogs["BagId"].max()
22      for i in range(1,result+1):
23          bag = dogs[dogs['BagId'] == i]
24          if (bag['Class'] == 1).any():
25              labels.append(float(1))
26          else:
27              labels.append(float(-1))
28          bag.drop(['Class', 'BagId'], axis=1)
29          bags.append(bag.values.tolist())
```

Cabe destacar que se comentará y descomentarán, aquellas líneas que hacen referencia a cada dataset.

Una vez se han preparado los datos, es necesario dividirlos en train y test, para comprobar el funcionamiento del modelo entrenado, el código es el siguiente:

```
train_bags, test_bags, train_labels, test_labels = train_test_split(bags, labels, stratify=labels)
```

Finalmente se invocan los modelos que se pretenden utilizar. Se entrena con dichos modelos y se realiza la predicción de los mismos para obtener resultados a comparar. El código es el siguiente:

```
    # Construct classifiers
    classifiers = {}
    classifiers['MissSVM'] = misvm.MissSVM(kernel='linear', C=1.0, max_iters=20)
    classifiers['sbMIL'] = misvm.sbMIL(kernel='linear', eta=0.1, C=1e2)
    classifiers['SIL'] = misvm.SIL(kernel='linear', C=1.0)
    print("Comienzan los entrenamientos:")
    # Train/Evaluate classifiers
    accuracies = {}
    for algorithm, classifier in classifiers.items():
        classifier.fit(train_bags, train_labels)
        predictions = classifier.predict(test_bags)
        accuracies[algorithm] = np.average(test_labels == np.sign(predictions))

    for algorithm, accuracy in accuracies.items():
        print('\n%s Accuracy: %.1f%%' % (algorithm, 100 * accuracy))
```

Los resultados obtenidos son los siguientes:

```
MissSVM Accuracy: 5.0%

sbMIL Accuracy: 97.4%

SIL Accuracy: 99.4%
```

2.2. Segunda parte:

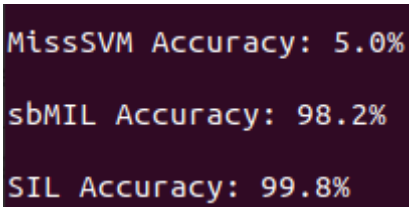
El dataset utilizado en este caso es el dataset **CorelFloower** con 7947 instancias, repartiéndose dichas instancias en 2000 bags distintas. El fichero consta de 11 columnas, siendo la primera la clase de cada patrón, y la segunda, la mochila a la que pertenece. Las 9 columnas restantes hacen referencia a los distintos atributos de cada patrón.

Una vez escogido el dataset, este se carga en el programa, aplicando el procesamiento igual que en el apartado anterior para todo el tema de las bags y las etiquetas.

El código realizado es similar al visto anteriormente.

Una vez se han preparado los datos, es necesario dividirlos en train y test, para comprobar el funcionamiento del modelo entrenado, igual que en el primer caso.

Finalmente se invocan los modelos que se pretenden utilizar, igual que en el primer caso. Los resultados obtenidos son los siguientes:



2.3. Tercera parte:

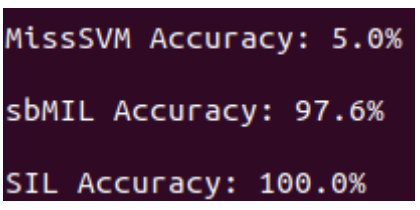
El dataset utilizado en este caso es el dataset **CorelFood** con 7947 instancias, repartiéndose dichas instancias en 2000 bags distintas. El fichero consta de 11 columnas, siendo la primera la clase de cada patrón, y la segunda, la mochila a la que pertenece. Las 9 columnas restantes hacen referencia a los distintos atributos de cada patrón.

Una vez escogido el dataset, este se carga en el programa, aplicando el procesamiento igual que en el apartado anterior para todo el tema de las bags y las etiquetas.

El código realizado es similar al visto anteriormente.

Una vez se han preparado los datos, es necesario dividirlos en train y test, para comprobar el funcionamiento del modelo entrenado, igual que en el primer caso.

Finalmente se invocan los modelos que se pretenden utilizar, igual que en el primer caso. Los resultados obtenidos son los siguientes:



3. Resultados y Conclusiones:

Para mayor claridad, se agrupan los resultados obtenidos en los experimentos en la siguiente tabla:

	MissSVM	sbMIL	SIL
CorelDogs	5.0%	97.4%	99.4%
CorelFlowers	5.0%	98.2%	99.8%
CorelFood	5.0%	97.6%	100%

Cabe destacar, que tanto el entrenamiento de MissSVM y sbMIL, conllevan tiempo de cómputo mucho más superiores al tiempo de cómputo del modelo SIL, siendo un aspecto interesante a tener en cuenta a la hora de su elección.

Comparando los resultados obtenidos de los modelos, en la tabla anterior, podemos apreciar que en general, el modelo MissSVM da muy malos resultados para estos datasets, probablemente se deba por la estructura de dichos datasets. Los modelos sbMIL y SIL dan mejores resultados. En concreto, como se comentó en la introducción, SIL suele funcionar generalmente bien a pesar de su relativa sencillez, de hecho en este caso, prácticamente clasifica correctamente todas las bags del test en los distintos dataset.

Respecto a los dataset, todos ellos son muy similares, ya que el número de instancias, bags, etc, son iguales. Por ende, los resultados obtenidos entre ellos son muy parejos.