Отчет по домашней работе №1

Предсказание появления ребер в графе с временными метками

Выполнил студент гр. 22М04 Беховский М. А.

Введение.

В данной работе были проведены исследования для четырех графов: UC (http://konect.cc/networks/opsahl-ucsocial/), Rado (http://konect.cc/networks/radoslaw_email/), bitA (http://konect.cc/networks/soc-sign-bitcoinalpha/), SX-MO (http://konect.cc/networks/sx-mathoverflow/).

В первой части работы будут представлены некоторые полученные сетевые характеристики.

Во второй будут приведены результаты экспериментов по предсказанию появления ребер в графе при помощи модели логистической регрессии на основании статических и темпоральных признаков.

I. Свойства графов.

Сетевые характеристики были посчитаны для статических вариантов графов (кратные ребра были удалены).

Характеристики для наибольшей компоненты связности при содержании в ней более 1000 вершин были оценены по принципу “снежный ком”: к случайно выбранной вершине КС добавлялись ее соседи и соответствующие ребра, пока мощность “снежного кома” не превысила 1000 вершин. Оценки получены осреднением по 10 “снежным комам”, *поэтому оценки являются вещественными числами*. Необходимо отметить, что данный метод с использованными параметрами не может обеспечить высокую точность оценки в случае, когда наибольшая КС графа по мощности значительно выше 1000.

Если же наибольшая компонента связности имела мощность, не превышающую 1000 по вершинам, характеристики были посчитаны напрямую.

	UC	Rado	SX-MO	bitA
Число вершин	1899	167	24759	3783
Число ребер	13838	3250	187986	14124
Плотность	0.008	0.234	0.001	0.002
Число компонент связности	4	1	45	5
Доля вершин в наибольшей компоненте связности (нкс)	0.997	1	0.996	0.998
метод оценивания характеристик нкс	снежный ком	прямое выч.	снежный ком	снежный ком
радиус нкс	5.8	3	2.8	5.1
диаметр нкс	11.3	5	4.8	9.8
90 процентиль расстояний	7.3	3	3.5	6.7
Средний	0.109	0.592	0.313	0.178

кластерный коэффициент				
Коэффициент ассортативности	-0.188	-0.295	-0.215	-0.169

II. Предсказание появления ребер в графах.

2.1 Выборка экземпляров

Источником примеров для тренировочно-тестовой выборки являлся срез графа, в котором были удалены ребра с временными метками, лежащими после 2/3 квантиля.

Из этого среза были выбраны пары вершин, находящиеся на расстоянии 2 друг от друга. Из полученного множества было выбрано 10000 пар, между которыми на последнем временном отрезке образуется ребро, и 10000 пар, между которыми ребро не образуется, для того чтобы в тренировочной выборке не было дисбаланса классов. Выборка пар проводится с заменой, так как не во всяком графе в множестве вершин на расстоянии 2 найдется 10000 пар вершин какого-либо класса (по естественным причинам чаще недобор возникает по классу соединенных в будущем вершин). Данная выборка из 20000 пар разделена стандартной функцией библиотеки sklearn на тренировочную и тестовую с параметрами по умолчанию.

2.2 Признаковое пространство

На данный момент реализовано три варианта выделения признаков: статический (4-х мерный вектор [CN, AA, JC, PA] без учета временных меток), темпоральный с учетом кратности ребер (присутствует этап агрегирования прошлых событий, в результате получается 96-мерный вектор признаков), темпоральный без учета кратности (отсутствует этап агрегирования, для взвешивания ребра берется последняя временная метка по кратным ребрам, в результате получается 12-мерный вектор признаков).

Признаки извлекаются из среза графа, данные из последней трети ребер не “просачиваются”.

Более подробное описание и объяснение размерности векторов можно наблюдать в прилагаемой программе.

Изначально предполагалось за базовый брать темпоральный вариант с учетом кратности для сравнения со статическим, а вариант без учета кратности использовать только в том случае, если во входном графе нет кратных ребер. Однако в текущей реализации извлечение признаков с агрегированием оказывается более трудоемким по сравнению с более простым вариантом без агрегирования настолько, что было принято решение провести эксперименты с вариантом без агрегирования. Руководство по запуску прилагаемой программы и выбору нужного извлечения признаков см. в приложении к отчету.

2.3 Модель машинного обучения

Предсказания вычислялись моделью логистической регрессии из пакета sklearn. Признаковые векторы предварительно нормализуются (обнуляется среднее и сводится к единице дисперсия). Для измерения качества результатов согласно заданию использовалась метрика ROC AUC, также вычисленная при помощи средств пакета sklearn.

2.4 Результаты экспериментов

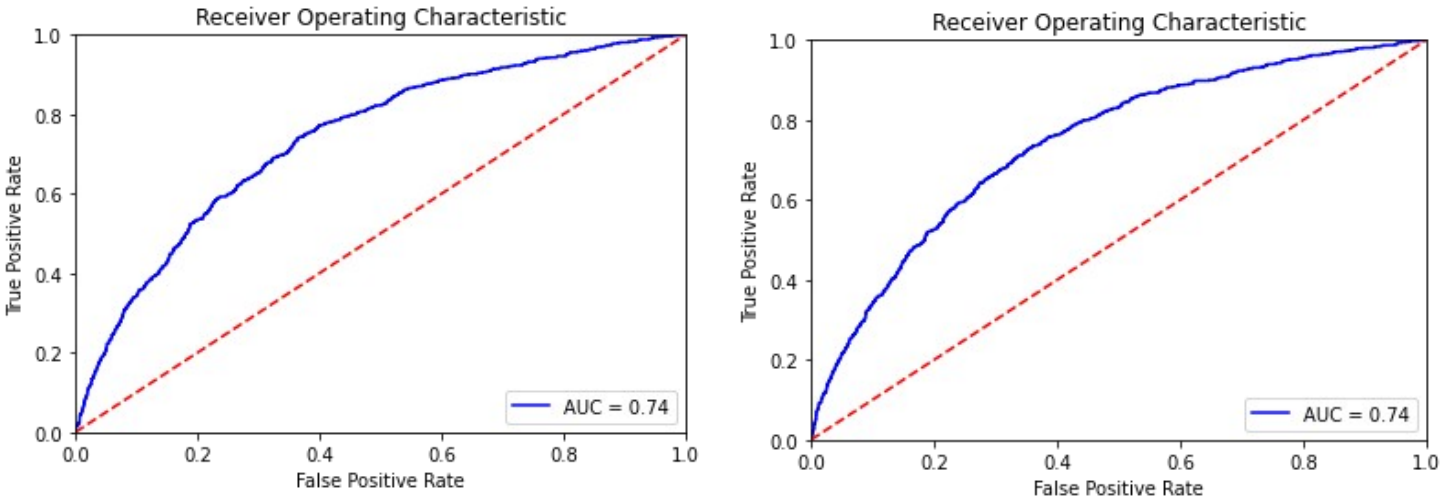


Рис. 2.4.1 ROC кривые для предсказаний со статическими признаками (слева) и темпоральными(справа) для графа UC

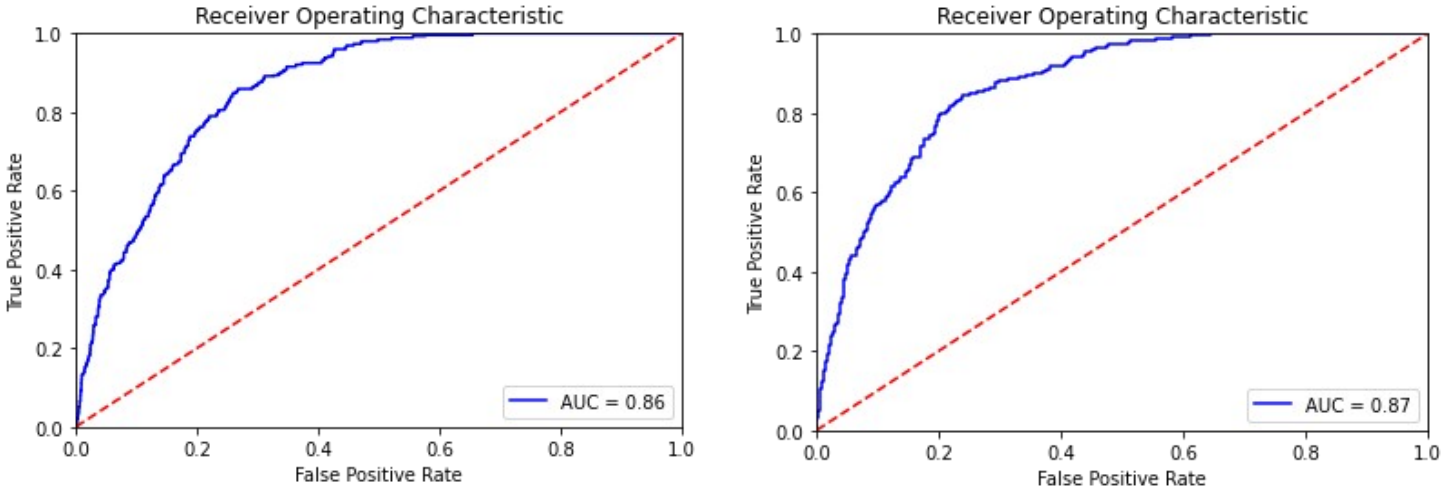


Рис. 2.4.2 ROC кривые для предсказаний со статическими признаками (слева) и темпоральными(справа) для графа Rado

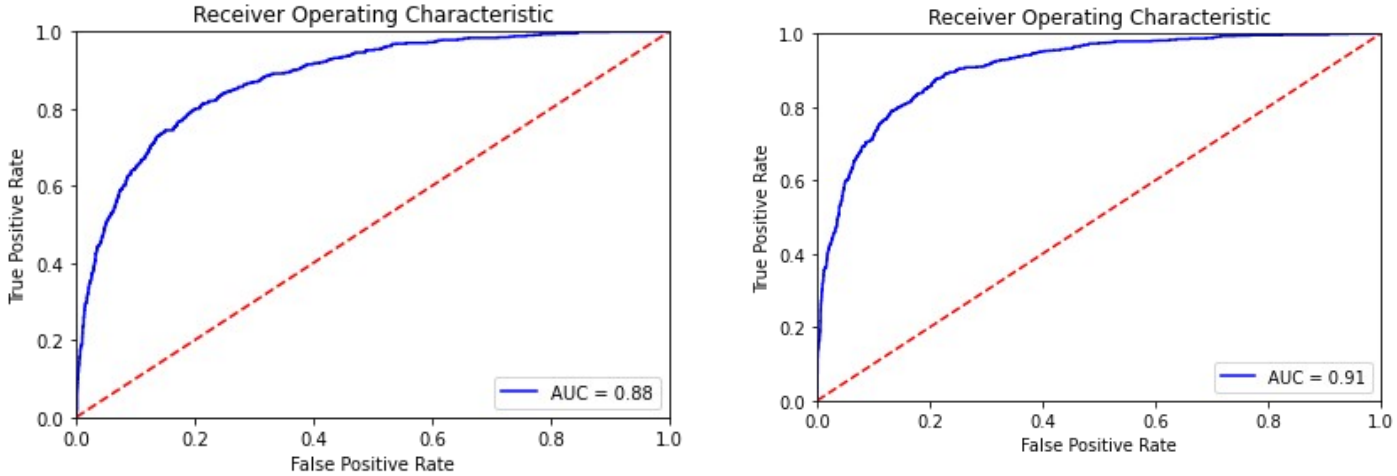


Рис. 2.4.3. ROC кривые для предсказаний со статическими признаками (слева) и темпоральными(справа) для графа bitA

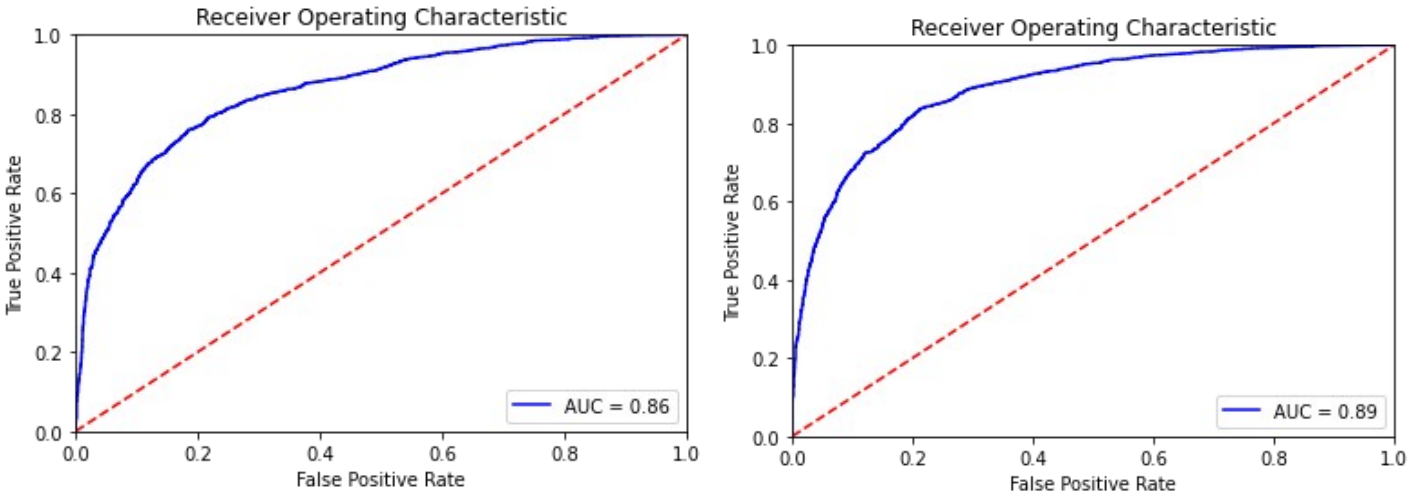


Рис 2.4.4 ROC кривые для предсказаний со статическими признаками (слева) и темпоральными(справа) для графа SX-MO

Осмелимся сделать вывод о большей точности при использовании темпоральных признаков на основании четырех экспериментов. Более подробную информацию, содержащую, в том числе, данные о времени выполнения программы, можно найти в самой программе.

3. Приложение

Работа в программе ведется через вызов методов класса GraphAnalyzer, в котором хранятся представления нужных графов (избыточные с целью повышения производительности по времени), признаковые векторы и прочая необходимая информация.

Для инициализации необходимо вызвать конструктор класса, обязательно подав в качестве аргумента путь к файлу, содержащему граф.

```
lab1 = GraphAnalyzer(fpath='data/out.opsahl-ucsocial')

Total number of edges 59835
100%|██████████| 1449/1449 [00:04<00:00, 341.30it/s]
```

рис 3.1 Пример создания среды для экспериментов с графом

Входной файл должен состоять исключительно из строк – ребер графа. В строке первые два символа – вершины ребра, четвертый символ – временная метка ребра, третий символ – вес ребра. Третий символ игнорируется. Пример входного файла приложен к отчету. Строки с метаданными (если такие есть до списка вершин) придется удалить.

При вызове конструктора происходит формирование среза графа для выбора обучающих примеров и формирование списка пар-кандидатов, что может оказаться довольно продолжительным процессом, так как для этого необходимо решать задачу нахождения кратчайших путей длины не больше 2, поэтому выводится прогресс исполнения цикла выбора пар вершин.

Для вывода характеристик статического графа используется метод explore_graph:

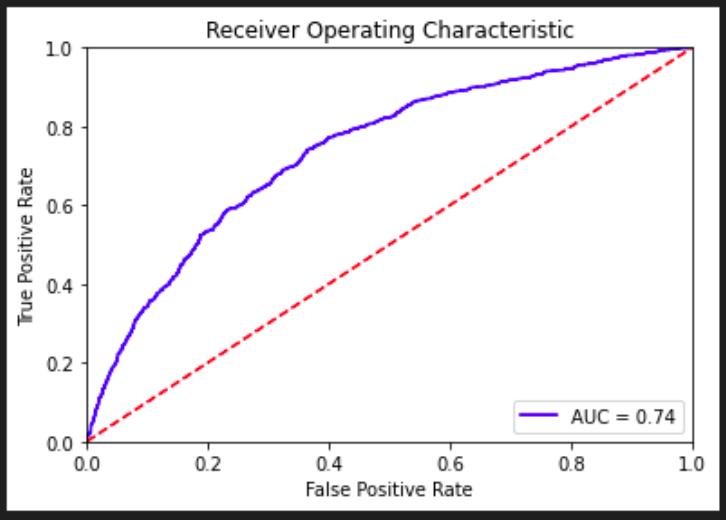
```
lab1.explore_graph()
```

```
 Статический граф:
 Вершин: 1899
 Ребер: 13838
 Плотность: 0.007678601848568738
 Компонент связности: 4
 Доля вершин в максимальной по мощности компоненте связности: 0.9968404423380727
Snowball
 Наибольшая компонента связности:
 1.Радиус 5.8
 2.Диаметр 11.3
 3.90 процентиль расстояния: 7.3
 Средний кластерный коэффициент: 0.10939892385364355
 Коэффициент ассортативности -0.18777578714668058
```

Для запуска обучения и решения задачи классификации используется метод obs (one button solution):

```
lab1.obs()
```

```
 Number of candidates 219633
 Number of candidates which will be connected 1027
 Number of candidates which will remain unconnected 218606
Example set is of size 20000
100%|██████████| 20000/20000 [00:00<00:00, 23713.91it/s]
static features calculated
```

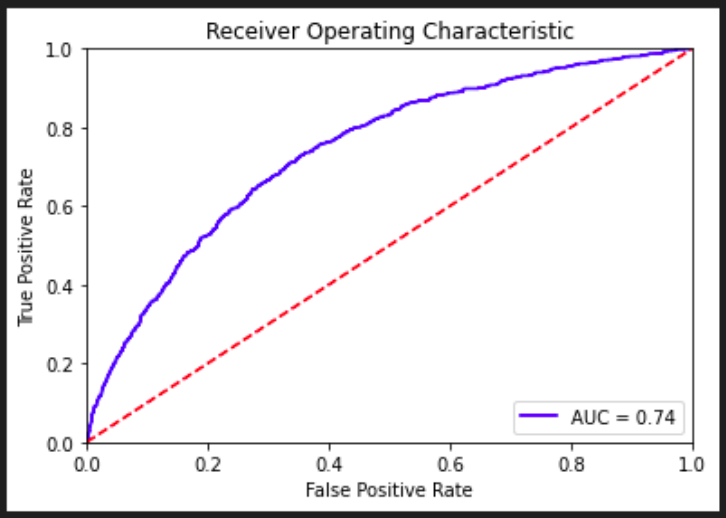



```
0.7396293362775218
```

В результате будут выведены некоторые данные по процессу сэмплирования и прогрессу вычисления векторов признаков. По умолчанию работа проводится со статическими признаками. Для работы с темпоральными надо методу obs в качестве аргумента подать ‘temporal’:

```
lab1.obs('temporal')
```

```
 Number of candidates 219633
 Number of candidates which will be connected 1027
 Number of candidates which will remain unconnected 218606
Example set is of size 20000
100%|██████████| 20000/20000 [00:10<00:00, 1943.68it/s]
temporal features calculated
```




```
0.7425832053251408
```

Если в графе на расстоянии два вершин значительно меньше 10000 и в выборке будет много повторов, признаки будут присвоены довольно быстро, так как они запоминаются на пару вершин и заново не вычисляются.

Если во входном графе нет мультиребер, то темпоральные признаки будут вычисляться без учета кратности ребер. Если же в графе мультиребра есть, можно вручную выключить/включить учет кратности ребер методом switch_persistence:

```
lab1.switch_persistence()
```

```
Warning. Manually switched to persistent mode. Computations will take significantly shorter
```

Программа выдаст предупреждение с указанием текущего режима извлечения темпоральных признаков (persistent игнорирует кратность ребер и берет последнюю доступную временную метку вместо агрегирования, discrete events проводит агрегирование).

Для проведения экспериментов на корректность работы программы на графах малой размерности с кратными ребрами метод switch_persistence рекомендуется не использовать. Признаки и так должны вычисляться достаточно быстро. Кроме того, в аргументы метода obs можно передать аргумент replace=False, чтобы сэмплирование производилось без замены, и в результате для обучения была создана сбалансированная выборка меньшей мощности:

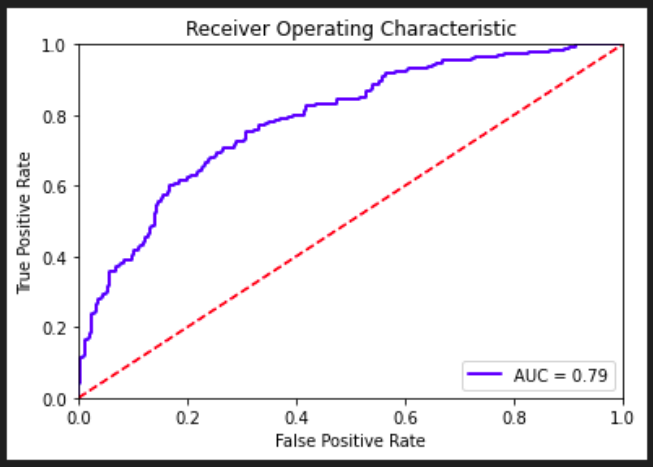
```
lab1.obs('temporal', replace=False)
```

✓ 1m 59.3s

```
 Number of candidates 219633
 Number of candidates which will be connected 1027
 Number of candidates which will remain unconnected 218606
Example set is of size 2054
100%|██████████| 2054/2054 [01:59<00:00, 17.24it/s]
temporal features calculated

/home/sleepydragon/.local/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:814:
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```




```
0.786581663630844
```

Необходимо отметить, что в некоторых случаях (см. картинку) обучение классификатора не сходится за число итераций, заданное по умолчанию. Тем не менее, результаты, полученные в “недообученной” модели оказываются лучше, чем результаты предсказаний на основе статических признаков.