ВМК

Задание 6. Использование Cython для решения задачи кластеризации

Практикум 317 группы, 2015

Начало выполнения задания: 10 марта 2015 года.
Срок сдачи: **23 марта 2015 года, 23:59.**
Максимальный балл: 5.0 (плюс бонусные баллы).
Текст задания обновлён 19 марта 2014 года. Изменился порядок столбцов в матрице points!

Содержание

1 Кратко о *Cython*

Cython — расширение для языка программирования *Python*, сочетающее в себе одновременно высокую производительность(сравнимую с языком программирования *C*), а также высокую выразительность(сравнимую с языком программирования *Python*). Почти любая программа, написанная на языке программирования *Python* может быть также скомпилирована на *Cython* в отдельный модуль, вызывая который в коде *Python* мы можем получить некоторое ускорение(около $10 - 20\%$) просто за счет того, что код оказался предварительно скомпилированным, а не вызывается поэлементно интерпретатором. Но это, очевидно, не единственный и не основной способ использования *Cython*. Производительность *Python-программы* можно заметно увеличить за счет того, что в самых трудозатратных кусках кода можно указать компилятору тип данных, который мы собираемся использовать и дальше опираться на вычисления именно с этим типом.

Один из самых простых примеров, показывающих каким образом можно добиться заметного ускорения — это итерирование по элементам *Python-списка*. Во-первых, каждый элемент *Python-списка* является максимально возможно «абстрактным» объектом и, помимо того, что требует для своего хранения больше памяти, он также требует достаточно затратные вычисления на то, чтобы понять, с каким типом сейчас будет производиться операция и только затем можно выполнять доступ к самим данным по ссылке на имеющийся «абстрактный» объект.

Кроме того, что *Cython* позволяет писать производительный код, являющийся смесью языков *Python* и *C*, он также позволяет оформлять код в модули, которыми можно пользоваться из *Python-программ*, а также использовать код, написанный непосредственно на языках *C* и *C++*.

Общая схема использования языка *Cython* обычно является примерно следующей.

1. Написать код на *Cython*, оформленный в виде файлов с расширением **.pyx*, а также(необязательно) код на *C/C++*, оформленный стандартным образом.

2. Запустить транслятор файлов **.pyx* в файлы на языке программирования *C*.

3. Скомпилировать и слинковать итоговый результат из *C-файлов*, полученный траслитерацией **.pyx*, а также нескольких *C/CPP-файлов*, код которых используется в файлах **.pyx* и, возможно, нескольких статических или динамических библиотек общего пользования, использованных в **.pyx/c/cpp*.

4. Использовать полученную динамическую библиотеку(файл с расширением **.so* в *UNIX-подобных ОС* или **.dll* в *ОС Windows*) с помощью команды *import* в *Python-программе* или *Python-модуле*.

Для более подробного изучения рекомендуется рассмотреть примеры кода, приложенные к заданию, а также самостоятельно прочитать некоторые основные моменты из документации, найденной в сети «Интернет».

2 Задание

В качестве задания вам предлагается реализовать два алгоритма кластеризации взвешенных точек на сфере на заранее заданное число кластеров *K-Means* и *K-Medoids* с использованием *Cython* и *C++*, сравнить их с реализациями этих же алгоритмов на *Python* по производительности. Если более детально, то нужно реализовать следующие 4 части.

1. Реализация метода *K-Means* на языке *Python*. Никаких требований по эффективности на эту реализацию не накладывается, но в случае эффективной работы можно получить дополнительные баллы. Допускается использование *NumPy* функций.

2. Реализаций метода *K-Medoids* на языке *Python*. Никаких требований по эффективности на эту реализацию не накладывается, но в случае эффективной работы можно получить дополнительные баллы. Допускается использование *NumPy* функций.

3. Эффективная реализация метода *K-Means* на языке *Cython*. Необходимо получить максимально эффективную реализацию, удовлетворяющую требованиям по качеству. Допускается использование *NumPy* функций.

4. Эффективная реализация метода *K-Medoids* на языке *C++*, а также обертка на *Cython* к этой реализации. Необходимо получить максимально эффективную реализацию, удовлетворяющую требованиям по качеству.

Кроме того, необходимо будет визуализировать точки на глобусе при помощи библиотеки *mpl_toolkits.basemap*. Более подробно о визуализации можно прочитать в разделе «Визуализация».

Подробное описание используемых алгоритмов легко найти в сети «Интернет». В качестве расстояния между точками нужно использовать кратчайшее расстояние по сфере, а не кратчайшее расстояние в трехмерном пространстве.

В алгоритме *K-Means* мы предполагаем, что вы будете использовать следующий метод построения центра кластера. Выберем две любые точки кластера и рассмотрим кратчайший путь между ними. Разобьем путь по длине на два пути пропорционально весам точек, и условно заменим две выбранные точки на новую получившуюся точку с весом, равным сумме весов двух точек. Будем повторять предыдущие шаги алгоритма, пока в кластере не останется одна точка, её мы и выберем центром.

Аналогично в алгоритме *K-Medoids* требуется при выборе точки, которая будет новым центром кластера, использовать в качестве критерия не сумму расстояний до всех остальных точек кластера, а сумму расстояний, умноженных на веса точек, до которых расстояния вычисляются.

Для сравнения производительности методов необходимо запускать фиксированное количество итераций, равное 50. Тест производительности должен производится на подвыборке соответствующей Северное Америке $continent = NA$ (см. «Данные»). Для каждого метода необходимо нарисовать график времени работы от количества точек, количества точек выбираются с шагом 100 и равновероятно сэмплируются из выборки соответствующей Северной Америке. Количество кластеров всегда равно количеству стран из Северной Америки во входных данных, то есть не зависит от подвыборки.

Кроме того, необходимо нарисовать график зависимости ускорения самой быстрой версии по отношению к реализации на чистом питоне от числа точек, для методов *K-Means* и *K-Medoids*.

Для самой быстрой версии *K-Means* и *K-Medoids* необходимо представить качество кластеризации всей входной выборки (см. «Оценка качества алгоритма») на количество кластеров, равное количеству стран во входных данных. Необходимо также визуализовать полученную кластеризацию (см. «Визуализация»). Кроме того, для каждого из методов, необходимо отдельно кластеризовать каждый из континентов, привести качество кластеризации внутри континента, а также качество кластеризации всей выборки, полученное таким способом. Полученную кластеризацию также необходимо визуализовать.

3 Данные

Исходные данные представлены в виде текстового файла *data.tsv* в следующем формате.

```
continent \t country \t city \t polulation \t longitude \t latitude
```

Где *continent* - двухбуквенное сокращение континента.
country - двухбуквенное сокращение страны.
city - название города
population - численность населения, которое вам необходимо использовать как вес города.
longitude - долгота
latitude - широта

4 Оценка качества алгоритма

Качество работы алгоритма будет проверяться при помощи 10 независимых запусков алгоритма, при этом, если хотя бы на половине запусков алгоритм покажет приемлемое качество, то он будет участвовать в конкурсе на производительность. При оценке производительности будет проводится N независимых запусков с усреднением времени по каждому запуску. В качестве функционала качества будет использоваться метод *Rand-Index*, равный доле пар точек выборки, для которых алгоритм и разметка одновременно отнесли точки к разным кластерам или одновременно отнесли к одному кластеру. Будет использоваться реализация *Rand-Index* из библиотеки *sklearn*, а именно *sklearn.metrics.adjusted_rand_score*. Качество будет оцениваться на скрытой выборке, совпадающей по размеру и способу выбора с обучающей выборкой в файле *data.tsv*. Количество кластеров равно количеству стран в обучающей выборке. Ориентировочным порогом на значение *Rand-Index* можно считать значение 0.16, однако проверяющие оставляют за собой право изменить этот порог в зависимости от результатов студентов.

5 Требования к реализации

Каждый из 4-х методов, реализующих алгоритмы, должен удовлетворять следующей сигнатуре.

```
def AlgoName(
        numpy.ndarray[float, ndim=2, mode='c'] points not None,
        int points_number,
        int clusters_number,
        numpy.ndarray[int, ndim=1, mode='c'] clusterization not None)
```

Это означает, что каждый из методов должен корректно отработать, если передать ему первым параметром *numpy.array* с типом данных *float*, размерности 2. При этом мы предполагаем, что в качестве входных данных передается массив размера $N \times 3$, где N — общее число точек и для каждой точки хранятся значения *weight*, *longitude* и *latitude*. Функция не должна изменять входные данные! Вторым параметром передается количество точек. Третьим параметром передается требуемое число кластеров. Четвертым параметром передается массив кластеризации, в котором для каждой точки нужно определить номер ее кластера(может быть произвольным целым числом, помещающимся в тип *int*). Этот массив должен заполнять сам алгоритм!

Данные требования не означают необходимость всегда жестко задавать сигнатуру. Например, *Python-функция*

```
def SlowKMeans(input, input_size, cl_number, result)
```

вполне может удовлетворять изложенным выше ограничениям, если она корректно отработает при передаче ей данных через *numpy.array*.

Подробнее о передаче в *Cython-методы* данных в виде *numpy.array* вы можете прочитать в примерах кода с комментариями, прилагающихся к этому заданию, а также в сети «Интернет».

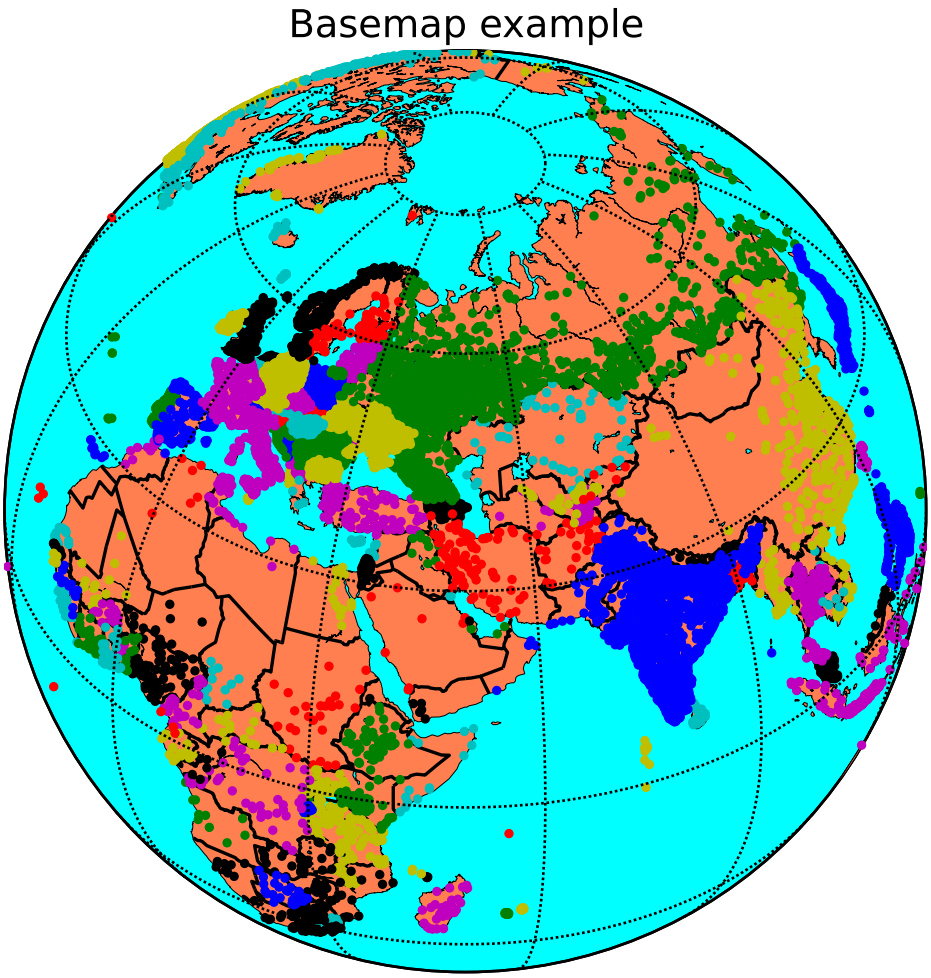
Результатом работы является директория с названием *StudentLastName_StudentFirstName*. Внутри директории должны быть только файлы, необходимые для компиляции проекта.

1. *Cython-методы* должны быть реализованы в файле *cy_cluster.pyx*
2. *Python-методы* должны быть реализованы в файле *py_cluster.py*.
3. Название методов должно в точности совпадать с *k_means* и *k_medoids*!
4. В директории должен присутствовать файл *setup.py*, который должен корректно собирать модуль *cy_cluster*.
5. В директории должен присутствовать файл *test.py*, в котором подключаются модули *cy_cluster* и *py_cluster* и выполняется простейшая генерация случайных данных и запуск всех 4-х алгоритмов.

6 Визуализация

Для визуализации точек на глобусе необходимо использовать библиотеку *mpl_ toolkits.basemap*, всю необходимую информацию, в том числе про установку, можно найти на сайте *http://matplotlib.org/basemap/index.html*. Работоспособность системы на ОС Windows не тестировалась, однако на Unix-like системы библиотека ставится достаточно просто. Визуализировать точки можно на карте, а не на глобусе, в случае глобуса ожидается несколько ракурсов, чтобы можно было увидеть все точки выборки.

В целом ожидается что-то вроде этого:



Где разными цветами обозначены различные кластеры.

Для отрисовки точек на карте можно воспользоваться методом *Basemap.plot* с указанием параметра *latlon=True*.

7 Сдача задания

Сдача задания осуществляется через репозиторий SVN. Результатом работы должна быть директория, содержащая всё то, что описано в требованиях к реализации + отчет в формате *pdf*. В отчете необходимо представить результаты замеров производительности разных методов, а также основные идеи, которые позволили ускорить алгоритм, сохранив достаточный уровень качества.

8 Бонусные баллы

- +0.5 балла за самую быструю реализацию любого из *Cython-алгоритмов*(или отставание от самой быстрой не более, чем на 5%), в случае, если не более 4-х участников получат отставание не более 5%. Если таких участников окажется больше 4-х, то все они получат +0.2 балла.

- +0.3 балла за самую быструю реализацию любого из *Python-алгоритмов*(или отставание от самой быстрой не более, чем на 5%), в случае, если не более 4-х участников получат отставание не более 5%. Если таких участников окажется больше 4-х, то все они получат +0.1 балла.

- Запрещено ускорять алгоритм *K-Medoids* использованием чего-либо принципиально другого, кроме расстояний между точками и весов точек. Если за участником будет замечено что-то подобное, то никаких бонусных баллов он точно не получит. Решение о нечестном ускорении может быть принято проверяющим субъективно и безапеляционно.

  Пример, чтобы понять, что имеется ввиду — можно разогнать *K-Medoids* за счет выкидывания половины точек из рассмотрения(если при этом требования по качеству алгоритм всё еще проходит), но нельзя ускорять алгоритм, заменяя его, например, на *K-Means*. Также нельзя использовать никакое другое расстояние между точками, кроме честного расстояния на сфере (вычисление этого расстояния можно оптимизировать сколько угодно, но при условии сохранения приемлемой точности).

  Также запрещено улучшать качество алгоритма путем использования каких-либо знаний о выборке в целом, кроме тех, что можно определить из выборки, которая дана вам для обучения.