ВМК

Задание 1. Изучение Python, NumPy, SVN

Практикум 317 группы, 2014

Начало выполнения задания: 15 сентября 2014 года.
Срок сдачи: **6 октября 2014 года, 23:59.**
Максимальный балл: 5.0 (плюс бонусные баллы).

Содержание

1 Задание **1**

2 Бонусные баллы **2**

3 Работа с системой SVN **2**

4 Задачи **2**

1 Задание

Данное задание направлено на освоение студентами языка Python, системы научных вычислений NumPy, а также системы контроля версий SVN.

1. Научиться работать с системой SVN. Проделать операции, описанные в разделе ниже.
2. Для каждой из задач:
   (a) Написать на Python + NumPy несколько вариантов кода различной эффективности. Должно быть не менее трёх вариантов, в том числе как минимум один полностью векторизованный вариант, и один вариант без векторизации. Третий вариант решения — на ваше усмотрение, например, это может быть наиболее хорошо читаемый способ решения, или частично векторизованный вариант. Все варианты решения одной задачи должны содержаться в отдельном Python модуле.
   (b) Сравнить в IPython Notebook при помощи `%timeit` скорость работы на нескольких тестовых наборах разного размера (минимум 3).
   (c) Проанализировать полученные данные о скорости работы разных реализаций.
   (d) Получить выводы.
3. Написать отчет о проделанной работе (формат PDF). Отчёт предлагается выполнять в системе LaTeX. Можно воспользоваться другой системой (например, Word), но за это будет начислен штраф в 1 балл. Полезные ссылки: [1] [2]. Считается, что отчёт был выполнен в LaTeX, если в репозитории есть .tex файл и исходники картинок, из которых можно собрать PDF файл с отчётом. Отчет должен включать в себя описание работы с системой контроля версий: скриншоты или текст из консоли, иллюстрирующие, как проводить основные действия, такие как обновление и коммит.
4. После выполнения задания написать преподавателю об этом. Никакие файлы отправлять не нужно, будет проверяться последняя ревизия из репозитория.

Советы по выполнению задания.

- Чтобы перезагрузить уже загруженный в IPython модуль, воспользуйтесь функцией `importlib.reload`.
- Сохранить результаты запуска `%timeit` в переменную можно так: `x = %timeit -o func(x)`.

2 Бонусные баллы

- +0.25 балла. Написанный код полностью соответствует style guide PEP 8 [3]. Часть требований можно проверить при помощи утилиты flake8 [4].
- +0.25 балла. Ко всем задачам присутствуют автоматические тесты, проверяющие совпадение результатов работы всех вариантов кода. Тесты должны использовать встроенный в Python фреймворк unittest. Краткое руководство по этому фреймворку: [5]. Обратите внимание на модель `numpy.testing`, облегчающий написание тестов для NumPy массивов.
- +0.25 балла. IPython Notebook с экспериментами хорошо оформлен, легко читается, содержит комментарии к экспериментам.
- +0.25 балла. Хорошая работа с системой контроля версий, логически разделённые коммиты.

3 Работа с системой SVN

1. Установить на компьютер клиент SVN. Под ОС Windows рекомендуется TortoiseSVN [6].
2. Получить по почте логин, пароль и путь к репозиторию.
3. Сделать checkout из репозитория в рабочую папку.
4. Задание, а также написание отчёта предлагается выполнять в рабочей папке.

В финальной версии репозитория, которая будет проверяться, должны находиться:

1. Python модули с кодом.
2. IPython Notebook (один или несколько) с экспериментами.
3. PDF файл с отчётом.
4. Если отчёт написан в системе LaTeX, исходники отчёта: .tex файл и исходники картинок, нужные для сборки отчёта.

Никаких лишних файлов (.log и т.п.) в финальной версии репозитория быть не должно. Удалите их из репозитория, если они туда попали.

Рекомендации по использованию репозитория (за несоблюдение баллы сниматься не будут):

1. Изменения в решениях разных задач должны находиться в разных коммитах.
2. Приветствуется большое число коммитов.
3. Все коммиты должны содержать краткое описание (commit message).
4. Лишние файлы можно добавить в ignore list клиента SVN.

Структура папок в репозитории (к примеру, выделять ли папки trunk, branch, tags) остаётся на ваше усмотрение.

Замечание. Если по техническим причинам не удаётся воспользуется удалённым репозиторием, допускается использование локального, см. краткое руководство по созданию и работе с локальным репозиторием для TortoiseSVN [7]. Если клиент требует выбрать тип репозитория, выбирайте fsfs. Для проверки задания нужно будет отправить преподавателю заархивированный репозиторий (не рабочую папку).

4 Задачи

Предполагается, что модуль numpy импортирован под названием np.

1. Подсчитать произведение ненулевых элементов на диагонали прямоугольной матрицы. Для X = np.array([[1, 0, 1], [2, 0, 2], [3, 0, 3], [4, 4, 4]]) ответ 3.
2. Дана матрица X и два вектора одинаковой длины i и j. Построить вектор np.array([X[i[0], j[0]], X[i[1], j[1]], ..., X[i[N-1], j[N-1]]]).
3. Даны два вектора x и y. Проверить, задают ли они одно и то же мультимножество. Для x = np.array([1, 2, 2, 4]), y = np.array([4, 2, 1, 2]) ответ True.

4. Найти максимальный элемент в векторе x среди элементов, перед которыми стоит нулевой. Для x = np.array([6, 2, 0, 3, 0, 0, 5, 7, 0]) ответ 5.

5. Дан трёхмерный массив, содержащий изображение, размера (height, width, numChannels), а также вектор длины numChannels. Сложить каналы изображения с указанными весами, и вернуть результат в виде матрицы размера (height, width). Считать реальное изображение можно при помощи функции `scipy.misc.imread` (если изображение не в формате png, установите пакет pillow: `conda install pillow`). Преобразуйте цветное изображение в оттенки серого, использовав коэффициенты np.array([0.299, 0.587, 0.114]).

6. Реализовать кодирование длин серий (Run-length encoding). Дан вектор x. Необходимо вернуть кортеж из двух векторов одинаковой длины. Первый содержит числа, а второй - сколько раз их нужно повторить. Пример: x = np.array([2, 2, 2, 3, 3, 3, 5]). Ответ: (np.array([2, 3, 5]), np.array([3, 3, 1])).

7. Даны две выборки объектов - X и Y. Вычислить матрицу евклидовых расстояний между объектами. Сравнить с функцией `scipy.spatial.distance.cdist`.

8. Реализовать функцию вычисления логарифма плотности многомерного нормального распределения Входные параметры: точки X, размер (N, D), мат. ожидание m, вектор длины D, матрица ковариаций C, размер (D, D). Разрешается использовать библиотечные функции для подсчета определителя матрицы, а также обратной матрицы, в том числе в невекторизованном варианте. Сравнить с `scipy.stats.multivariate_normal(m, C).logpdf(X)` как по скорости работы, так и по точности вычислений.

Замечание. Можно считать, что все указанные объекты непустые (к примеру, в задаче №1 на диагонали матрицы есть ненулевые элементы).

Полезные функции NumPy: np.zeros, np.ones, np.diag, np.eye, np.arange, np.linspace, np.meshgrid, np.random.random, np.random.randint, np.shape, np.reshape, np.transpose, np.any, np.all, np.nonzero, np.where, np.sum, np.cumsum, np.prod, np.diff, np.min, np.max, np.minimum, np.maximum, np.argmin, np.argmax, np.unique, np.sort, np.argsort, np.bincount, np.ravel, np.newaxis, np.dot, np.linalg.inv, np.linalg.solve.

Многие из этих функций можно использовать так: x.argmin().

Список литературы

[1] К.В. Воронцов. Полезная информация для пользователей LaTeX. http://www.ccas.ru/voron/latex.html

[2] К.В. Воронцов. LaTeX2e в примерах. http://www.ccas.ru/voron/download/voron05latex.pdf

[3] PEP 8 — Style Guide for Python Code. http://legacy.python.org/dev/peps/pep-0008/

[4] Flake8. https://pypi.python.org/pypi/flake8

[5] Corey Goldberg. Python Unit Testing Tutorial. http://cgoldberg.github.io/python-unittest-tutorial/

[6] TortoiseSVN. http://tortoisesvn.net/

[7] Краткое руководство по работе с TortoiseSVN с локальным репозиторием. http://thinkinging.com/2007/04/12/creating-a-local-subversion-repository-with-tortoisesvn/