***Образец титульного листа:***

Федеральное государственное автономное образовательное учреждение

высшего образования

«Национальный исследовательский университет «Высшая школа экономики»

Факультет компьютерных наук

Образовательная программа Прикладная математика и информатика

бакалавриат

**01.03.02 Прикладная математика и информатика**

**О Т Ч Е Т**

**по учебной практике**

Выполнил студент гр. БПМИ185

Цыганов Артём Николаевич

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*(подпись)*

**Проверил:**

|  |  |  |
| --- | --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| *должность, ФИО руководителя от НИУ ВШЭ* | *оценка по 10 бальной шкале* | *подпись* |

**Москва, 2019**

**Содержание отчета**

|  |  |
| --- | --- |
| Календарный план-график с отметками о выполнении | Стр. 2 |
| Введение | Стр. 2 |
| Основная часть | Стр. 2 |
| Заключение | Стр. 2 |
|  | Стр. 3 |
| Список использованных источников | Стр. 3 |

**Календарный план-график с отметками о выполнении**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Введение**

Тензоры – это обобщение матриц на многомерный случай. Они получили распространение в последние годы в различных областях науки, например, в физике и математике. Основная проблема, с которой сталкиваются при работе с тензорами – экспоненциальный рост памяти по отношению к количеству размерностей. Именно поэтому ведется поиск новых, более удобных и эффективных, способов представления тензоров. Одним из таких способов является tensor train decomposition, или же просто ТТ-формат. Основная идея данного метода – небольшая потеря точности (которую может задать сам пользователь) при существенной (для маленьких точностей) экономии памяти.

Изучением данного формата я и занимался в ходе практики.

**Основная часть**

**Задачи**

1. Прочитать статью Ивана Оселедеца [1], разобраться с основными понятиями.
2. Запрограммировать основные алгоритмы, приведённые в статье
3. Провести численные эксперименты

**Реализация алгоритмов и проведение численных экспериментов.**

Для реализации алгоритмов был выбран язык *Python 3*, так как в нём есть встроенные библиотеки, удобные при работе с методами линейной алгебры.

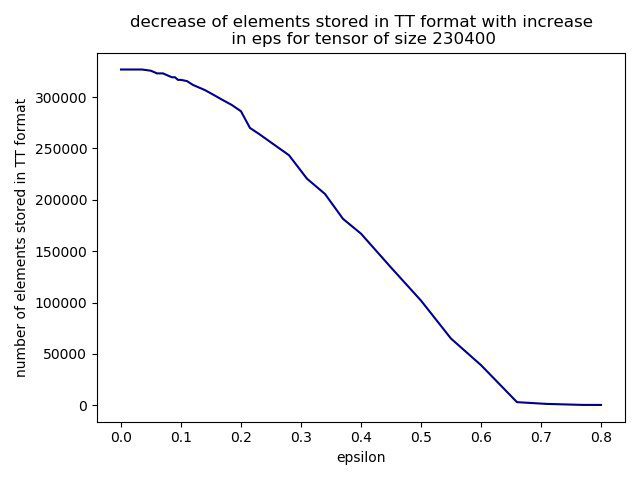
В самом начале написания класса *Tensor Train* был реализован алгоритм перевода тензора из его обычного представления (*numpy array*) в ТТ-формат (*TT-SVD*). Для реализации потребовались знания, полученные в курсе линейной алгебры и геометрии, например – SVD-разложение матриц. Перевод в ТТ-формат осуществляется с заранее заданной пользователем точностью . Как описано в статье, при правильной реализации данного алгоритма выполняется следующее неравенство:

|| A – B || ,

где В – тензор, которым мы приближаем исходный.

Первый эксперимент был направлен на то, чтобы показать, какой выигрыш по памяти мы получаем при различном значении эпсилон для тензора достаточно большого размера.

Для этого был сгенерирован случайный тензор размера 230400, а затем он был переведён в ТТ-формат с различной точностью. Результаты представлены на графике ниже.

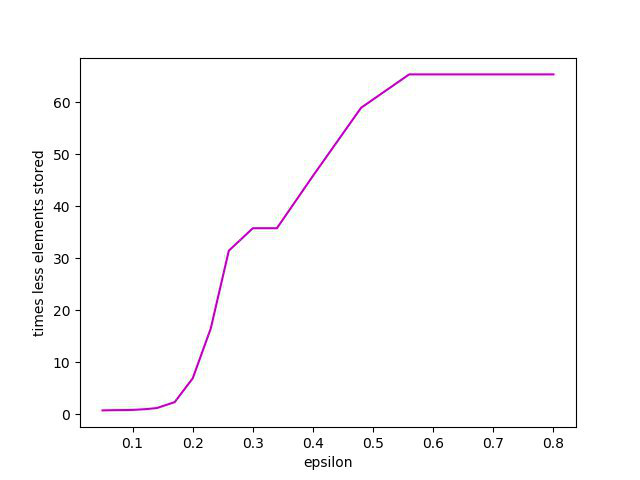


Далее были реализованы алгоритм для сложения двух тензоров в ТТ-формате, алгоритм для вычисления нормы Фробениуса у тензора, находящегося в ТТ-формате, а также алгоритм *dot-product*.

Последним и одновременно самым сложным для реализации алгоритмом стал *TT-rounding*. Как было описано выше, при помощи алгоритма TT-SVD можно добиться выигрыша в памяти при небольшой потере в точности. Но если тензор уже находится в ТТ-формате, то для работы данного алгоритма нужно будет вернуть его в первоначальное состояние, что является затратным (с точки зрения времени) процессом. Именно для избежания таких ситуаций придуман алгоритм *TT-rounding*.

Особой является ситуация, когда тензор уже находится в ТТ-формате, но не с совсем оптимальными рангами. Такое может случиться при выполнении различных операций (например, сложения) с тензорами в ТТ-формате. Чтобы не хранить лишней информации, но и не терять в точности, используют алгоритм *TT-rounding*. Подробное его описание есть в статье.

Для того, чтобы показать, насколько меньше элементов требуется хранить при применении данного алгоритма, был проведен следующий эксперимент. Было сгенерировано около 200 случайных тензоров, каждый из которых имел от 3 до 6 размерностей размера от 4 до 7. Данные тензоры были переведены в ТТ-формат с высокой точностью (0.0001), а затем к ним был применен алгоритм *TT-rounding* с различными эпсилон. Для того, чтобы график был более наглядным, для каждого эпсилон выбиралась медиана среди значений (размер до / размер после). Результаты представлены на графике ниже.



**Заключение**

В ходе практики я столкнулся с несколькими проблемами. Первая из них возникла еще на этапе прочтения статьи – к сожалению, в ней допущены различные опечатки, в связи с чем приходилось тратить гораздо больше времени на разбор конкретных теорем.

Следующие проблемы появились на этапе реализации алгоритмов.

Одной из целью практики было показать, что с ТТ-форматом можно работать быстрее, чем с обычным представлением тензоров. Однако в связи с тем, что библиотека *numpy* написана на языке *С++*, который является очень производительным, а алгоритмы из статьи были реализованы на *Python 3*, то убедиться в скорости не получилось. Мне кажется, если написать класс на языке *С++* и провести численные эксперименты, то ТТ формат окажется более производительным.

Также хотелось бы отметить, что более реалистичное представление о скорости работы возможно было бы получить на тензорах с очень большим числом элементов (несколько миллионов). Но, к сожалению, у меня не было возможности провести их, так как для этого нужен более мощный компьютер.

Тем не менее, эксперименты показывают, что данный формат действительно очень полезен. Отчетливо виден выигрыш в памяти, что существенно при работе с, например, облачными технологиями. Думаю, поиски новых способов представления тензоров будут и в дальнейшем.



**Описание полученных результатов**

Результатом является класс *Tensor Train*, содержащий в себе все необходимые методы. Код класса можно посмотреть по ссылке

**Список использованных источников**

[1] – Ivan Oseledets, Tensor-Train Decomposition

<https://www.researchgate.net/publication/220412263_Tensor-Train_Decomposition>