Подготовка отчётов. LaTeX. Markdown

ММП ВМК МГУ

Составлено на основе материалов Находнова Максима

Преподаватель: Алексеев Илья

- зачем нужны эксперименты
- как делать практикум
- как оформлять графики
- LaTeX
- Markdown

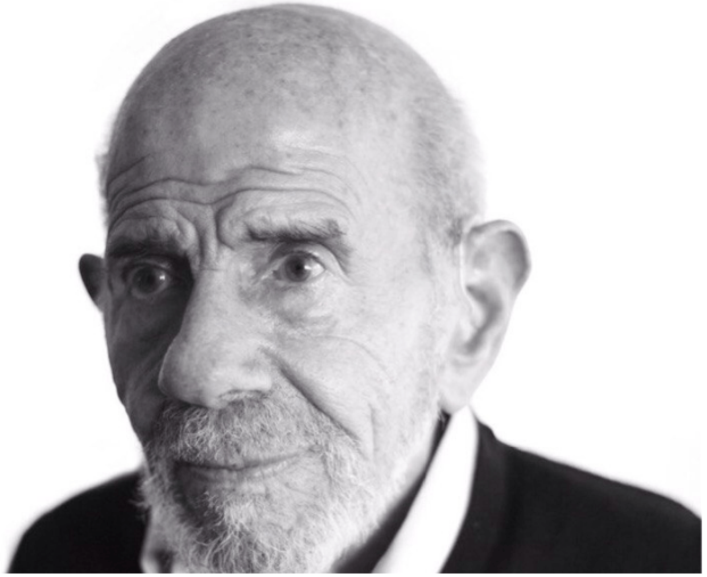
Цели больших практических заданий

- образовательная
- кодерская
- научная
- курсовая, диплом

Процесс выполнения практического задания

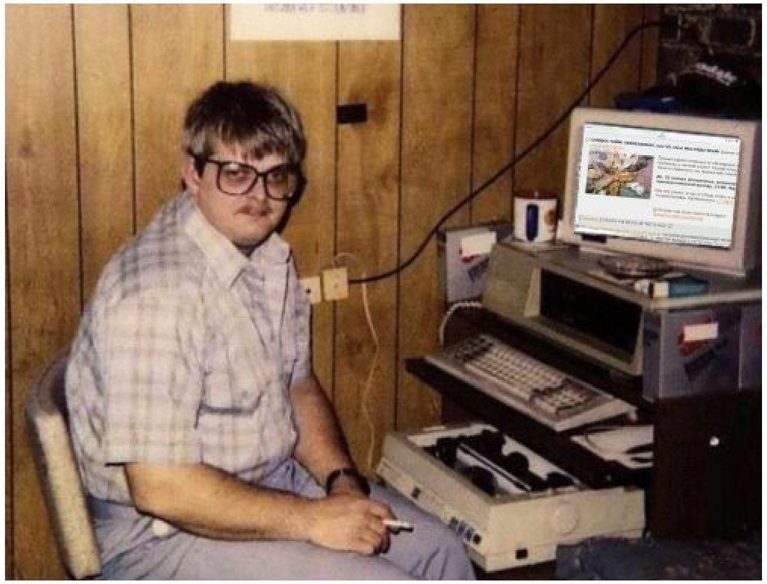
Этап 0: задуматься

- постановка задачи
- описание метода
- формулирование гипотез



Этап 1: реализовать метод

- выбрать среду разработки
- написать понятный код
- не забывать коммитить



Этап 2: посчитать эксперименты

главные проблемы:

- воспроизводимость
- сохранение прогресса

Сохранить всю сессию

```
from google.colab import drive
drive.mount('/content/drive')

import dill
dill.dump_session('/content/drive/MyDrive/prac1/session.db')
dill.load_session('/content/drive/MyDrive/prac1/session.db')
```

Сохранить отдельный объект: `pickle`

```
import pickle

with open('desired/path.pickle', 'wb') as handle:
        pickle.dump(model, handle, protocol=pickle.HIGHEST_PROTOCOL)

model = pickle.load(open('desired/path.pickle', 'rb'))
```

Не загружайте пиклы от непроверенных авторов!

Сохранить отдельный объект: `json`

```
import json

json.dump(logs, open('desired/path.json', 'w'))
json.load(open('desired/path.json', 'r'))
```

Сохранить `np.array`

```python
import numpy as np

np.save('desired/path.npy', my_nd_array)
my_nd_array = np.load('desired/path.npy')
```

Не загружайте массивы от непроверенных авторов!

Как делать перебор гиперпараметров

пример ноутбука

Этап 3: подготовка отчёта

зачем нужен отчёт?

- поделиться с миром тем, что вы придумали
- обеспечить воспроизводимость результатов
- в целом проанализировать и понять для себя что вы сделали

Раздел: заголовок (титульник)

сделано:

- что (отчет)
- кем (студент)
- где (ммп вмк мгу)

Раздел: введение

- обозначить предметную область
- общими и устоявшимися терминами описать задачу
- без копипасты!
- без математических формул

Плохое введение

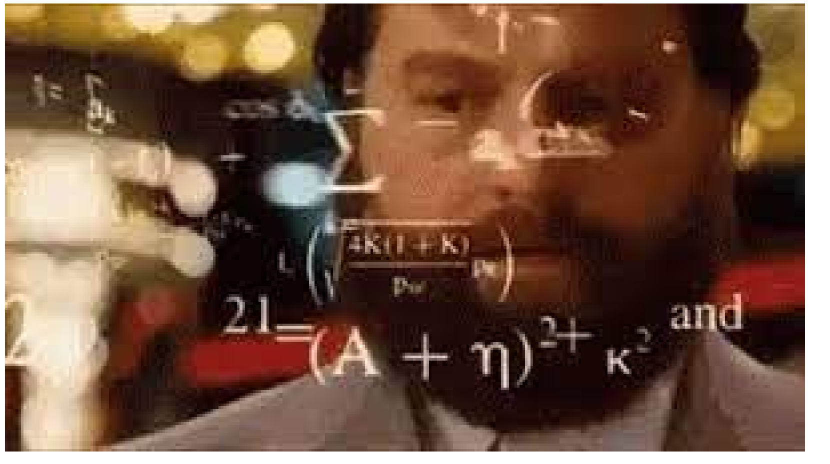
Компьютеры все чаще и чаще выступают помощниками человека. Водитель, потеряв дорогу, скорее воспользуется навигатором, чем спросит путь у прохожего или другого водителя. Захотев связаться с кем-то, мы скорее напишем ему письмо, состоящее из байтов, чем из бумаги и чернил. То же самое можно сказать и о распознавании изображений. Возьмем, как пример, ЕГЭ. Множество учеников со всей России заполняют огромное количество бланков, и все эти бланки необходимо проверить. И здесь на помощь приходят компьютеры. Они распознают ответы учеников и заполняют по ним базу данных, откуда берутся данные для подсчета результата экзамена. Теме анализа изображения, а точнее, анализа рукописного текста, и посвящена данная работа.

Хорошее введение

Данное практическое задание посвящено исследованию градиентного спуска и стохастического градиентного спуска на примере обучения логистической регрессии в задаче распознавания токсичности текста. Целью исследования является рассмотрение зависимости между сходимостью методов и величиной шага (learning rate), степенью затухания шага и начальным приближением весов модели. Также рассматривается влияние различных способов векторизации текста.

Раздел: пояснения к задаче

- формулы
- подробное объяснение методов, проблемы



Раздел: эксперименты

Главные цели этого раздела

- показать полученные вами результаты
- обеспечить их воспроизводимость другими людьми

Для этого пишем

- подробно (почти алгоритмично), но без кода
- иллюстративно
- с выводами / интерпретацией

Задание 2

Векторизированный вариант:

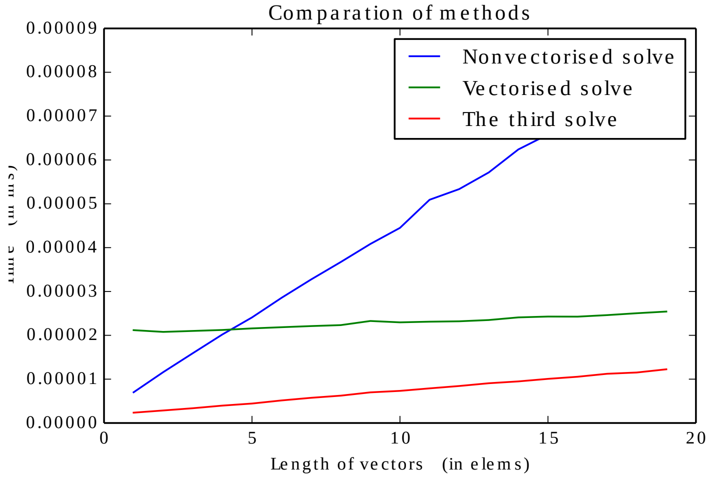
```
def vect_2(x, i, j):
    return np.vectorize(lambda z, y: x[z, y])(i, j)
```

Не векторизированный вариант:

```
def vect_1(x, i, j):
    r = np.array([], x.dtype)
    for k in range(np.size(i)):
        r = np.append(r, x[i[k], j[k]])
    return r
```

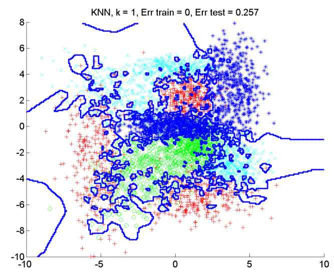
Третий вариант:

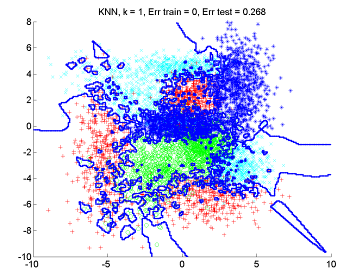
```
def vect_3(x, i, j):
    return np.array([x[t[0]][t[1]] for t in zip(i, j)])
```



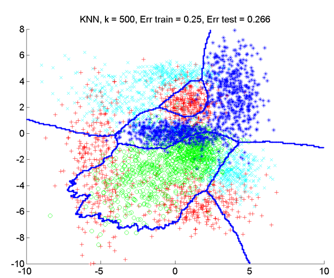
Самое оптимальное третье решение.

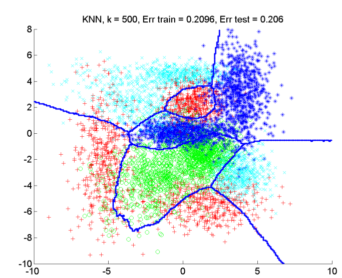
однако совершенно не приближается к оптимальному классификатору (уровень ошибки на тесте фиксирован на уровне 0.25). Увеличение объёма данных не приведёт к успеху в этой слишком простой модели, уровень ошибки существенно не уменьшится. Вот результат для 3000 и 5000 обучающих объектов:





Метод 500 ближайших соседей сильно недообучен, границы, выдаваемые им, слишком просты. Он обладает высоким *bias* (близкие кривые обучения на высоком уровне ошибки) при низком *variance*. При размере данных меньше 500 (50%) на тестовой выбоке наблюдается плато на уровне 75%: классификатор выдаёт в качестве ответа максимальный класс. На тестовой выборке при этом ошибка всё же ниже: сказывается случайный порядок объектов в обучающей выборке, соотношение классов не по 25%, и доминирующий класс досталяет чуть более 25%. Однако видна тенденция к падению уровня ошибки. Это происходит потому, что с увеличением объёма данных ослабляется эффект «доминирующего класса», и в этом случае добавление новых данных приведёт к значительному улучшению. Результат для 3000 обучающих объектов будет выглядеть гораздо более приемлемо, а для 5000 он даже приближается к оптимальному:





Налицо усложнение вида границ чисто за счёт большего объёма обучающей выборки. Это ещё раз наглядно иллюстрирует, насколько разный результат может давать одно и то же значение структурного параметра в зависимости от размера выборки.

3.2 SVM

Вот как выглядит зависимость ошибки SVM на обучении, валидации и тесте от каждого из параметров C и γ (другой параметр при этом фиксирован и равен 1; для параметров используется логарифмическая шкала):

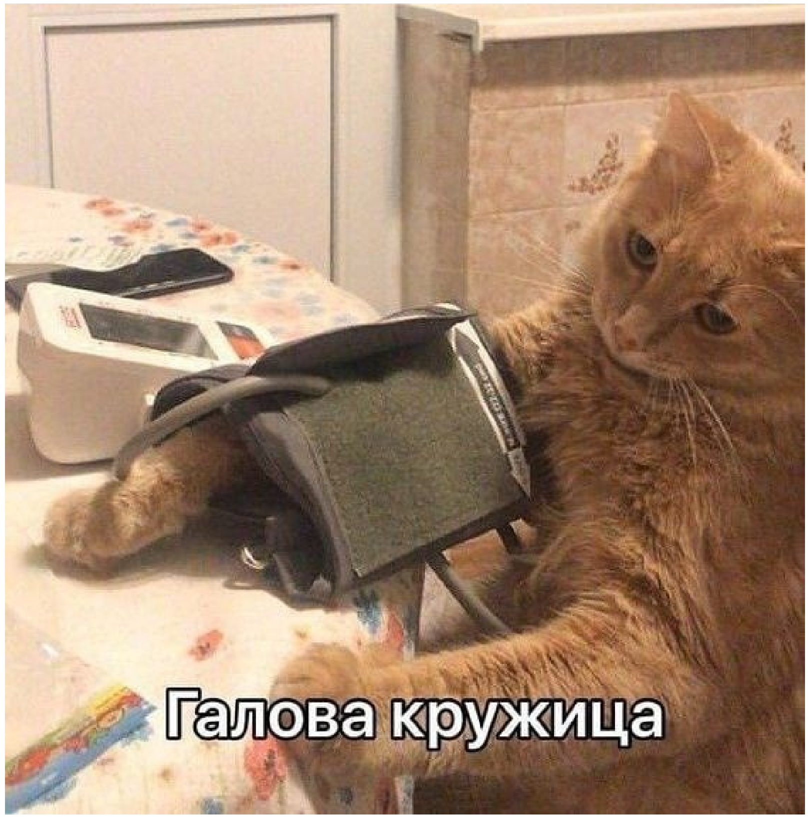
Раздел: заключение

- summary
- основные результаты
- отдельные удивительные моменты

Раздел: аппендиксы

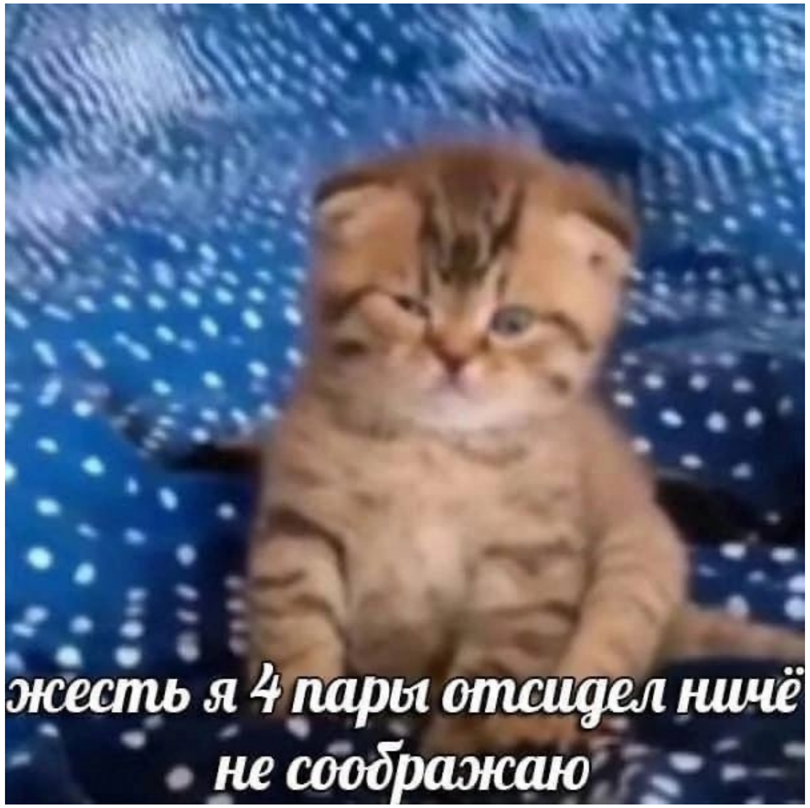
добавить дополнительный материал:

- листинг кода
- графики
- инженерные подробности
- листинг кода
- вспомогательные теоремы и утверждения



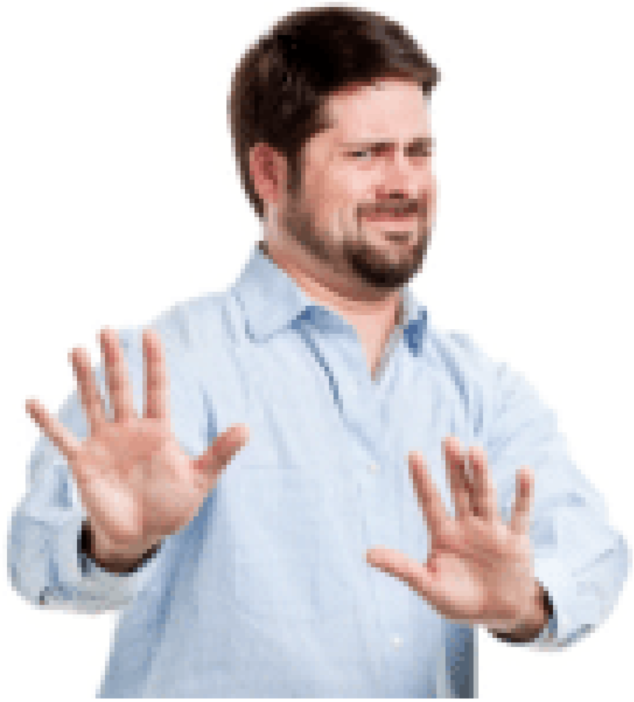
Структура отчёта

- титульник
- оглавление
- введение
- пояснение к задаче (opt)
- эксперименты (постановка, результаты, выводы)
- выводы / интерпретация
- заключение / общие выводы
- библиография
- аппендиксы (opt)



Чего стоит избегать

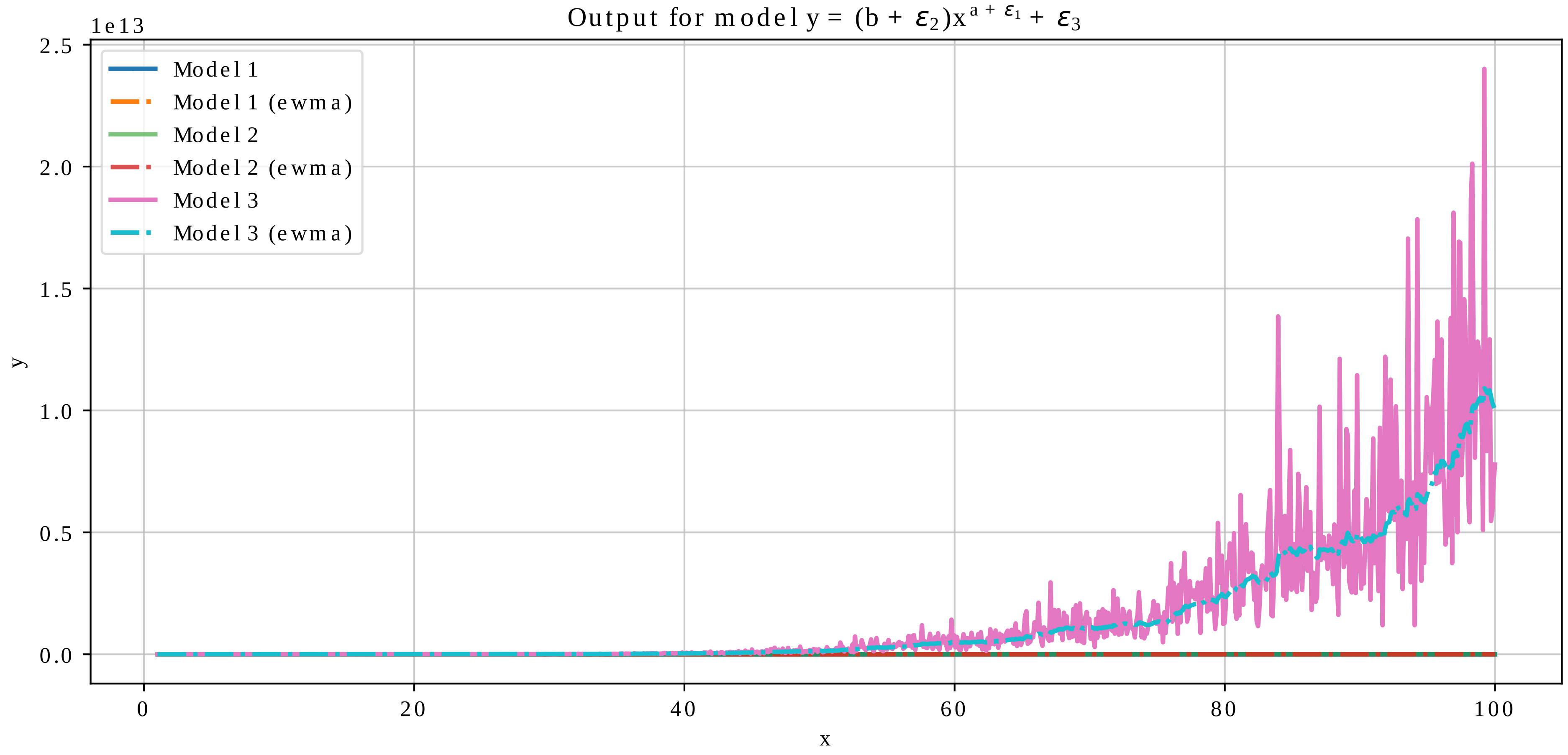
- безконкретных фраз ("результаты получились хорошими")
- ненаучной лексики ("результаты получились фиговыми")
- повествования от первого лица ("я пришел к выводу")
- обращений к читателю ("вашему вниманию представлены")
- грамматических ошибок

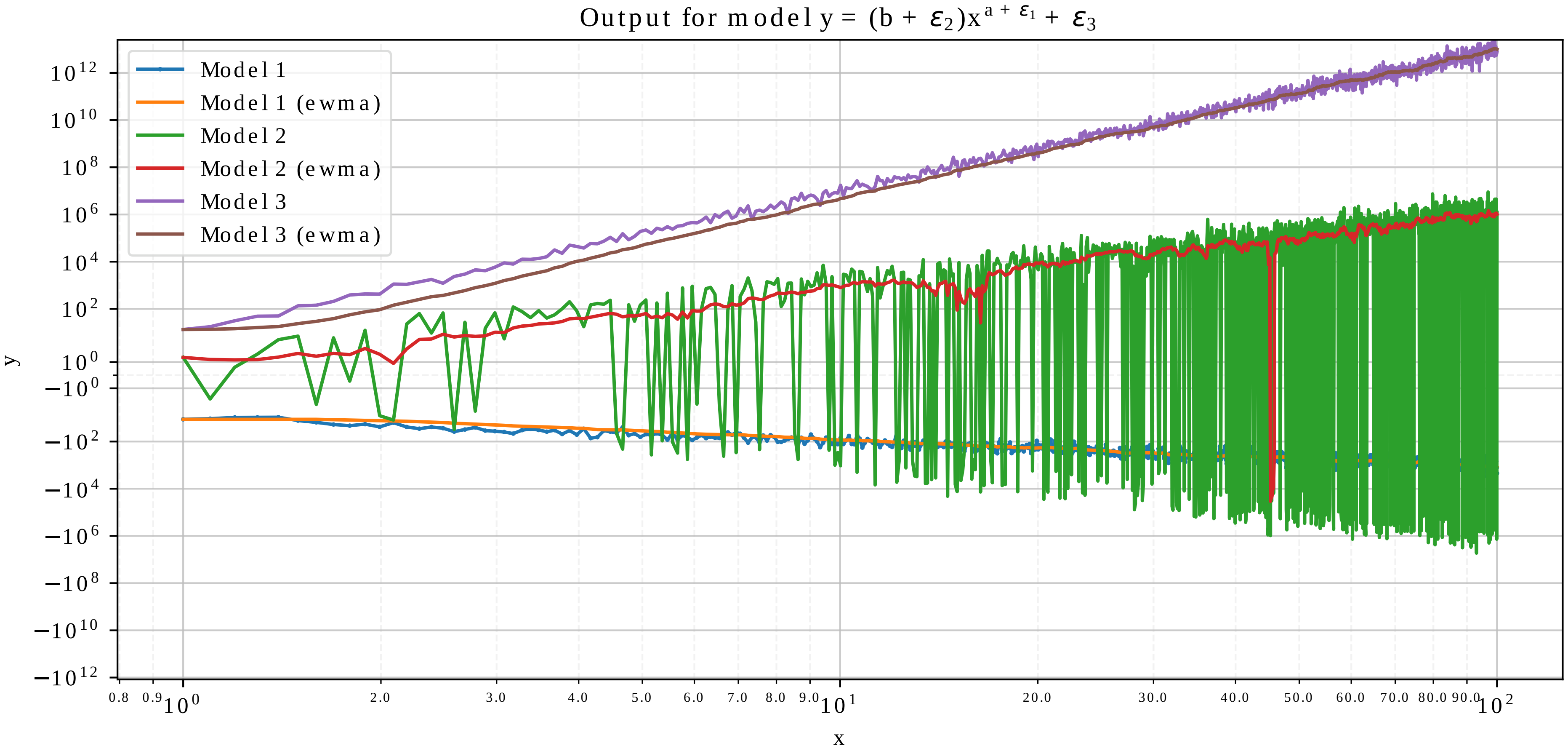


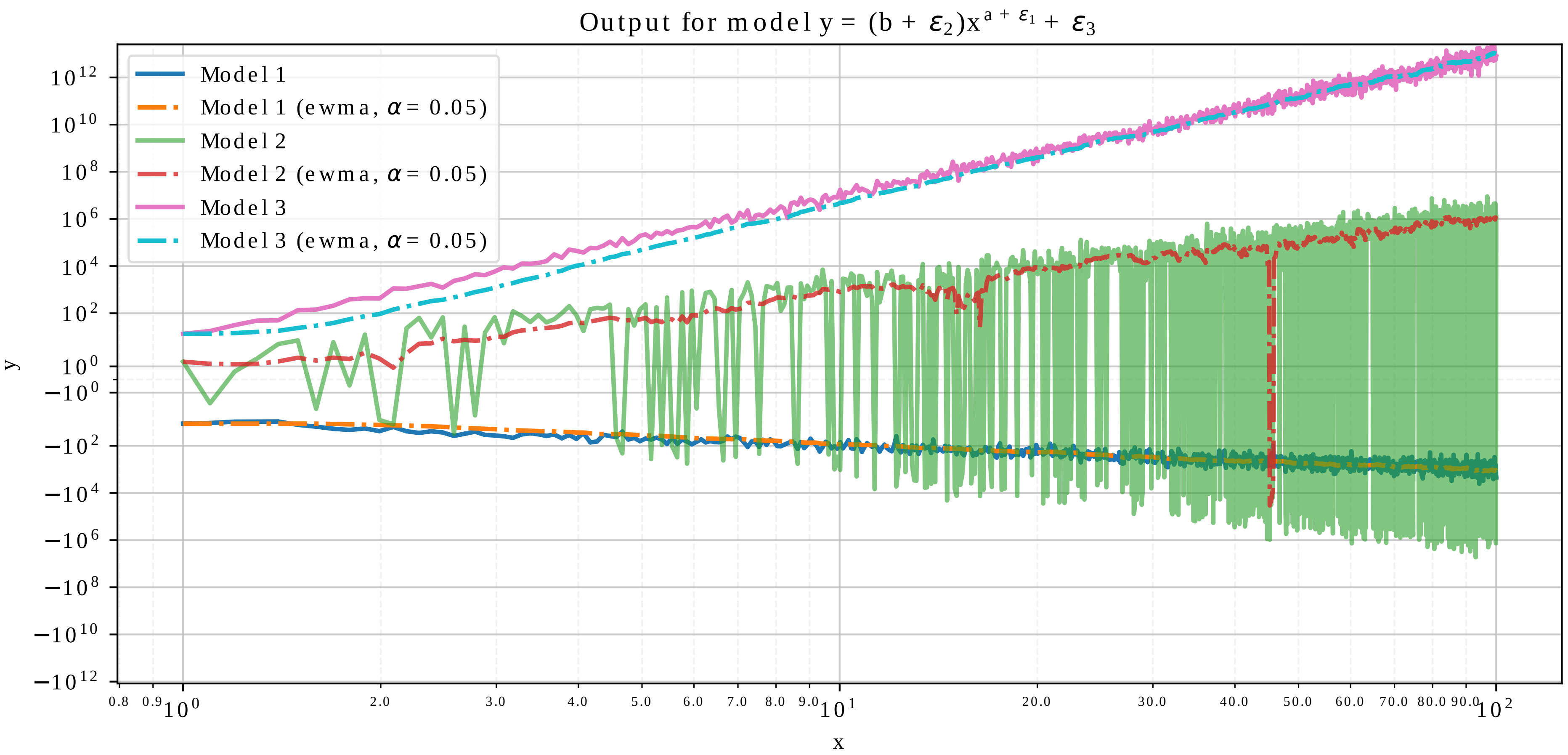
Итог: выполнение практикума

- Этап 0: задуматься
- Этап 1: реализовать метод
- Этап 2: провести эксперименты
- Этап 3: подготовка отчёта

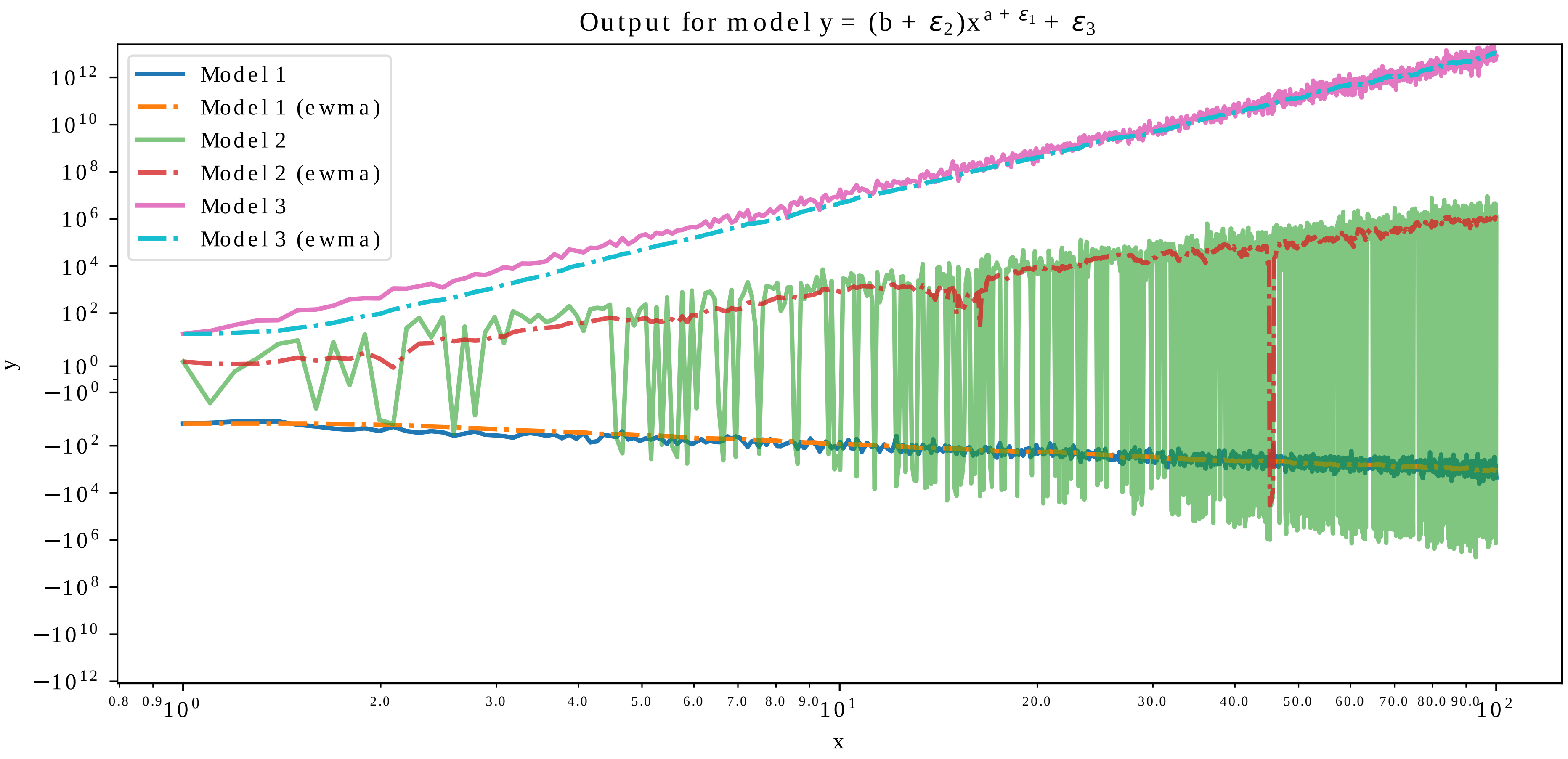
Оформление графиков


Output for model $y = (b + \varepsilon_2)x^{a + \varepsilon_1} + \varepsilon_3$
1e13
2.5
2.0
1.5
1.0
0.5
0.0
y
0
20
40
60
80
100
x
Model 1
Model 1 (ewma)
Model 2
Model 2 (ewma)
Model 3
Model 3 (ewma)


Output for model y = (b + ε2)x^(a + ε1) + ε3
Model 1
Model 1 (ewma)
Model 2
Model 2 (ewma)
Model 3
Model 3 (ewma)
y
x
10^12
10^10
10^8
10^6
10^4
10^2
10^0
−10^0
−10^2
−10^4
−10^6
−10^8
−10^10
−10^12
0.8
0.9
10^0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10^1
20.0
30.0
40.0
50.0
60.0
70.0
80.0
90.0
10^2



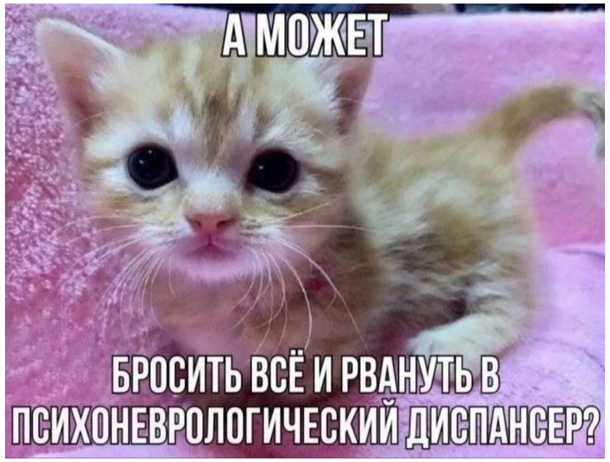
Output for model y = (b + ε2)x^(a + ε1) + ε3
Model 1
Model 1 (ewma, α = 0.05)
Model 2
Model 2 (ewma, α = 0.05)
Model 3
Model 3 (ewma, α = 0.05)
y
x



Output for model y = (b + ε2)x^(a + ε1) + ε3
Model 1
Model 1 (ewma)
Model 2
Model 2 (ewma)
Model 3
Model 3 (ewma)
y
x
10^12
10^10
10^8
10^6
10^4
10^2
10^0
−10^0
−10^2
−10^4
−10^6
−10^8
−10^10
−10^12
0.8
0.9
10^0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10^1
20.0
30.0
40.0
50.0
60.0
70.0
80.0
90.0
10^2

Неочевидные рекомендации

- делайте графики красивыми!
- текст на графиках такого же размера, что и в отчете
- подбирайте графики соответственно вашему типу данных (lineplot, scatterplot, barplot, violaplot)



Оформление страниц

2 THE GUMBEL-SOFTMAX DISTRIBUTION

We begin by defining the Gumbel-Softmax distribution, a continuous distribution over the simplex that can approximate samples from a categorical distribution. Let z be a categorical variable with class probabilities $\pi_1, \pi_2, ...\pi_k$. For the remainder of this paper we assume categorical samples are encoded as k-dimensional one-hot vectors lying on the corners of the $(k-1)$-dimensional simplex, Δ^{k-1}. This allows us to define quantities such as the element-wise mean $\mathbb{E}_p[z] = [\pi_1, ..., \pi_k]$ of these vectors.

The Gumbel-Max trick (Gumbel, 1954; Maddison et al., 2014) provides a simple and efficient way to draw samples z from a categorical distribution with class probabilities π:

$$z = \texttt{one\_hot}\left(\arg\max_i \left[g_i + \log \pi_i\right]\right) \tag{1}$$

where $g_1...g_k$ are i.i.d samples drawn from Gumbel$(0, 1)$[1]. We use the softmax function as a continuous, differentiable approximation to $\arg\max$, and generate k-dimensional sample vectors $y \in \Delta^{k-1}$ where

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(\pi_j) + g_j)/\tau)} \qquad \text{for } i = 1, ..., k. \tag{2}$$

The density of the Gumbel-Softmax distribution (derived in Appendix B) is:

$$p_{\pi,\tau}(y_1, ..., y_k) = \Gamma(k)\tau^{k-1} \left(\sum_{i=1}^{k} \pi_i / y_i^{\tau}\right)^{-k} \prod_{i=1}^{k} (\pi_i / y_i^{\tau+1}) \tag{3}$$

This distribution was independently discovered by Maddison et al. (2016), where it is referred to as the concrete distribution. As the softmax temperature τ approaches 0, samples from the Gumbel-Softmax distribution become one-hot and the Gumbel-Softmax distribution becomes identical to the categorical distribution $p(z)$.

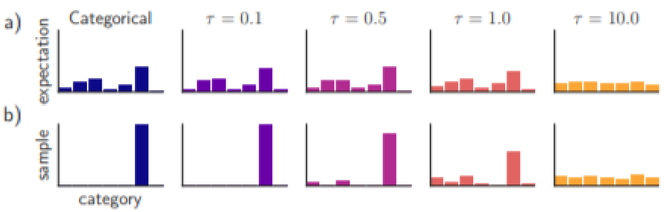


Figure 1: The Gumbel-Softmax distribution interpolates between discrete one-hot-encoded categorical distributions and continuous categorical densities. (a) For low temperatures ($\tau = 0.1, \tau = 0.5$), the expected value of a Gumbel-Softmax random variable approaches the expected value of a categorical random variable with the same logits. As the temperature increases ($\tau = 1.0, \tau = 10.0$), the expected value converges to a uniform distribution over the categories. (b) Samples from Gumbel-Softmax distributions are identical to samples from a categorical distribution as $\tau \to 0$. At higher temperatures, Gumbel-Softmax samples are no longer one-hot, and become uniform as $\tau \to \infty$.

2.1 GUMBEL-SOFTMAX ESTIMATOR

The Gumbel-Softmax distribution is smooth for $\tau > 0$, and therefore has a well-defined gradient $\partial y/\partial \pi$ with respect to the parameters π. Thus, by replacing categorical samples with Gumbel-Softmax samples we can use backpropagation to compute gradients (see Section 3.1). We denote

[1]The Gumbel$(0, 1)$ distribution can be sampled using inverse transform sampling by drawing $u \sim$ Uniform$(0, 1)$ and computing $g = -\log(-\log(\mathrm{u}))$.

Таблица 9: Результаты экспериментов задачи №8. Время работы измерено в микросекундах

	vectorized_8	non_vectorized_8	my_method_8	multivariate_normal
shape = (50, 100)	940.2	630531.5	42079.1	426.7
shape = (100, 200)	3985.8	4968530.7	386522.0	2636.1
shape = (150, 300)	10492.1	16607857.2	1408267.6	8813.9

Таблица 10: Результаты точности вычисления

	vectorized_8	non_vectorized_8	my_method_8
shape = (50, 100)	2.29e-13	1.29e-12	3.23e-12
shape = (100, 200)	2.44e-13	1.33e-12	3.29e-12
shape = (150, 300)	2.32e-13	1.29e-12	3.25e-12

Заметим, что стандартная реализациия scipy.stats.multivariate_normal отстает от vector_method:

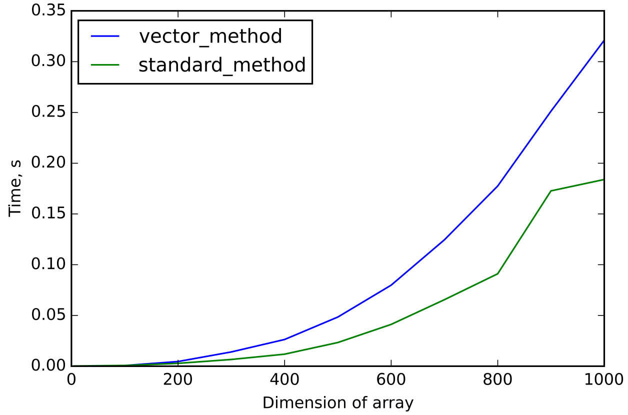


Рис. 6

Погрешность была вычислена как евклидова норма от разности двух функций:

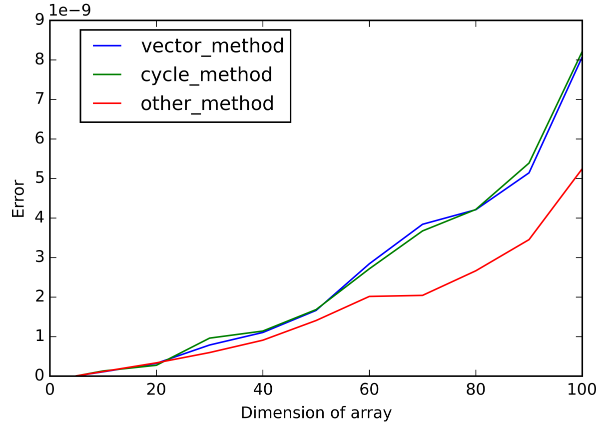


Рис. 7: Погрешность

Измерение времени

Время выполнения функций проверялось на квадратных матрицах X размера $n \times n$ сгенерированных из равномерного распределения и векторах i, j размера n сгенерированных из дискретного равномерного распределения. Результаты (Рис. 2), как и в задаче 1, показывают, что стандартные циклы в Python работают медленнее чем функции и методы, реализованные в библиотеке numpy. Индексация в numpy массивах работает медленнее чем метод take, но разница небольшая. В отличии от первой задачи, разница во времени выполнения между реализациями с ростом размерности данных не изменяется.

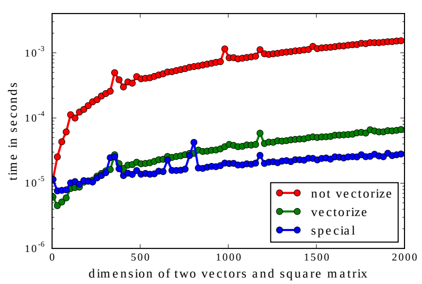


Рис. 2: Зависимость времени выполнения задачи 2 от размерности данных

LaTeX: полезные ссылки

- что мы затехали на паре: https://www.overleaf.com/read/mhnbpfgcdsby
- облачный компилятор: https://www.overleaf.com/
- поиск символа по рисунку: https://detexify.kirelabs.org/classify.html
- справочник: http://www.ccas.ru/voron/download/voron05latex.pdf

Markdown: полезные ссылки

- суперский редактор: https://typora.io/
- двухминутный ролик: https://youtu.be/KPSNiXQLc7I?si=EIOsruBjBRlppvyt