

# **Занятие 5.1: Написание текстовых отчётов и научных текстов**

**Практикум на ЭВМ, осень 2022**

Находнов Максим Сергеевич

кафедра ММП, ВМК МГУ

06 октября 2022 г.

## Написание научных текстов

*Научные тексты:* научные статьи, отчеты, курсовые, выпускные работы и т.д.

Есть правила (стандарты) написания научных текстов.

**Зачем нужны правила?**

Чтобы разные люди без дополнительных усилий понимали друг друга. Чем жёстче требования к терминологии, языку, форме подаче материала, оформлению, тем быстрее читатель сможет понять основную суть работы.

На этой паре разбираем, как писать отчёты.

# Типичная структура отчёта

Структура отчёта примерно соответствует пунктам задания:

1. **Титульная страница** (Заголовок)
2. (опционально) **Оглавление**
3. **Введение**
4. (опционально) **Пояснения к задаче**, например, выкладки для основных формул в работе
5. **Список экспериментов**
  - 5.1 Дизайн эксперимента
  - 5.2 Результаты эксперимента
  - 5.3 Выводы из эксперимента
6. **Общие выводы** из работы
7. (опционально) **Список использованной литературы**
8. (опционально) **Апpendиксы**, например, вспомогательные выкладки, пояснения

# У отчёта должен быть заголовок

Из заголовка должно быть понятно:

- ▶ Кем выполнено задание?
- ▶ Какое задание выполнено?
- ▶ К какому курсу относится задание?

Необязательно верстать титульную страницу!

## У отчёта должно быть **введение**

Из введения должно быть понятно:

- В чём заключалось задание?

Введение не повторяет полностью формулировку задания, а лишь кратко описывает её.

Можно ссылаться на формулировку задания, если не получается кратко сформулировать все аспекты.

Введение должно быть конкретным (относится к конкретному заданию, а не к выбранной области анализа данных в целом).

## Пример неконкретного введения к заданию

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. Что же делать в этой ситуации? Ведь вряд ли кто согласится проверять эти бланки. 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.

# Описание экспериментов: как не надо делать

## Задание 2

Векторизированный вариант:

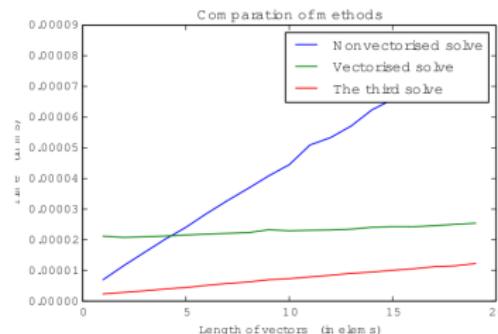
```
def vect_2(x, i, j):
    return np.vectorize(lambda x, y: x[z, y])(i, j)
```

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)])
```

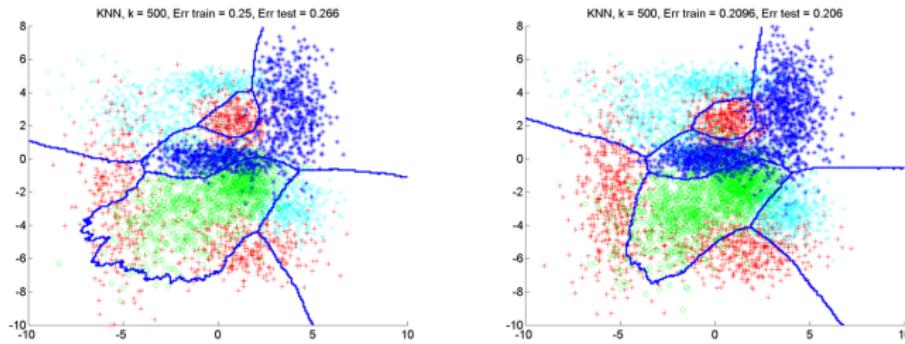


Самое оптимальное третье решение.

Эксперименты не описаны, нет выводов.

# Описание экспериментов: как надо делать

Метод 500 ближайших соседей сильно недообучен, границы, выдаваемые им, слишком прости. Он обладает высоким *bias* (блзкие кривые обучения на высоком уровне ошибки) при низком *variance*. При размере данных меньше 500 (50%) на тестовой выборке наблюдается пла-то 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 состаляет 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:



Проведён анализ и на его основе сделаны выводы.

## Анализируйте результаты экспериментов

размер данных	numpy	only python	смешанная
10	1.2	50.21	2.1
100	2.52	40.1	10.2
1000	4.5	45.34	30.95

Что в этих результатах подозрительно?

## Анализируйте результаты экспериментов

размер данных	pintpy	only python	смешанная
10	1.2	50.21	2.1
100	2.52	40.1	10.2
1000	4.5	45.34	30.95

Что в этих результатах подозрительно?

- ▶ Вычисления для размерности 100 и 1000 происходят быстрее чем для 10.

# Используйте векторные шрифты

## Растровые шрифты:

Для каждой задачи написать два различных варианта и один вариант работы реализаций на нескольких языках. Проанализировать полученные реализации и сделать выводы.

## Векторные шрифты:

Для выполнения задания было предложено задачи представлены в отдельных файлах (задачи).

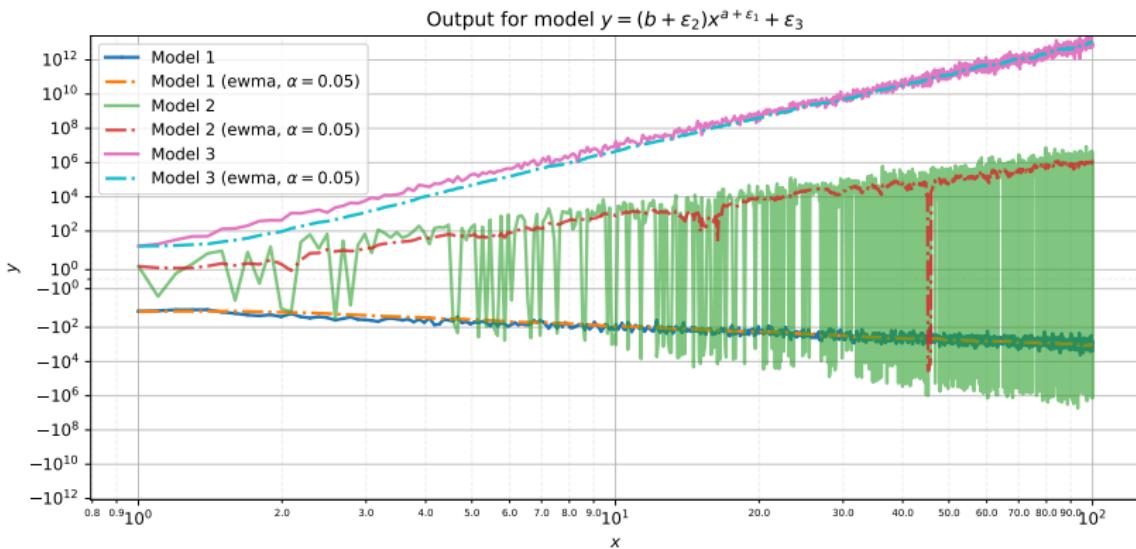
Реализация тестов представлена в модуле `test` образом: первая часть тестов проверяет корректность решения задачи, вторая часть проверяет совпадение решений.

Проведение экспериментов реализовано в модуле `experiment`, направленный на сравнение времени выполнения. Проводился 100 раз, после чего выбиралось наилучшее решение.

Для генерации текстов с векторными шрифтами достаточно установить пакет `TEX cm-super`.

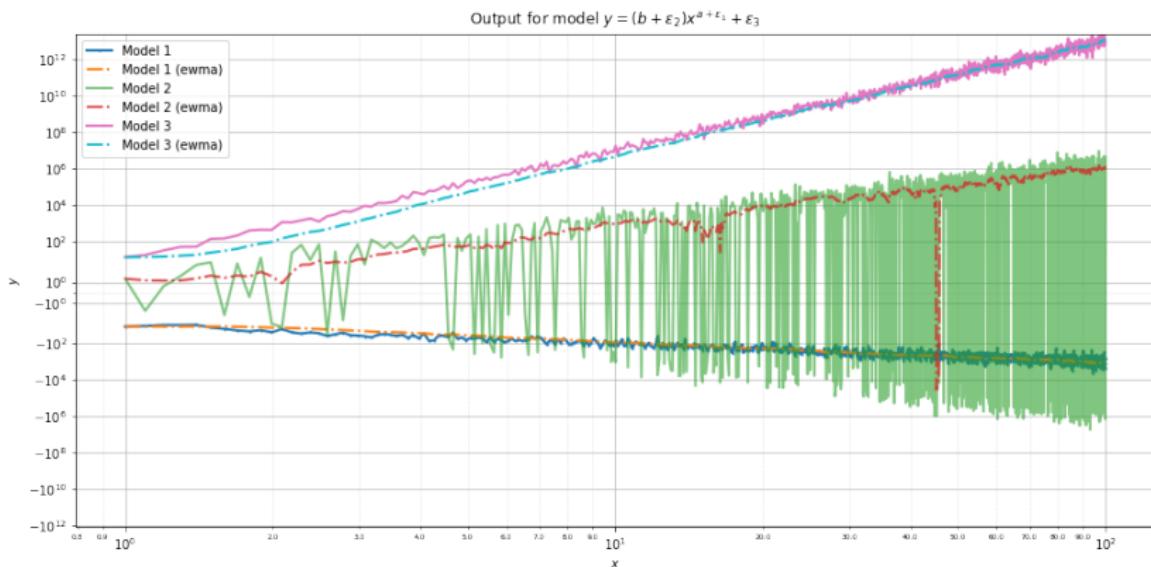
# Оформление графиков. Некоторые советы

Графики должны быть с одной стороны понятными и информативными, а с другой стороны красивыми.



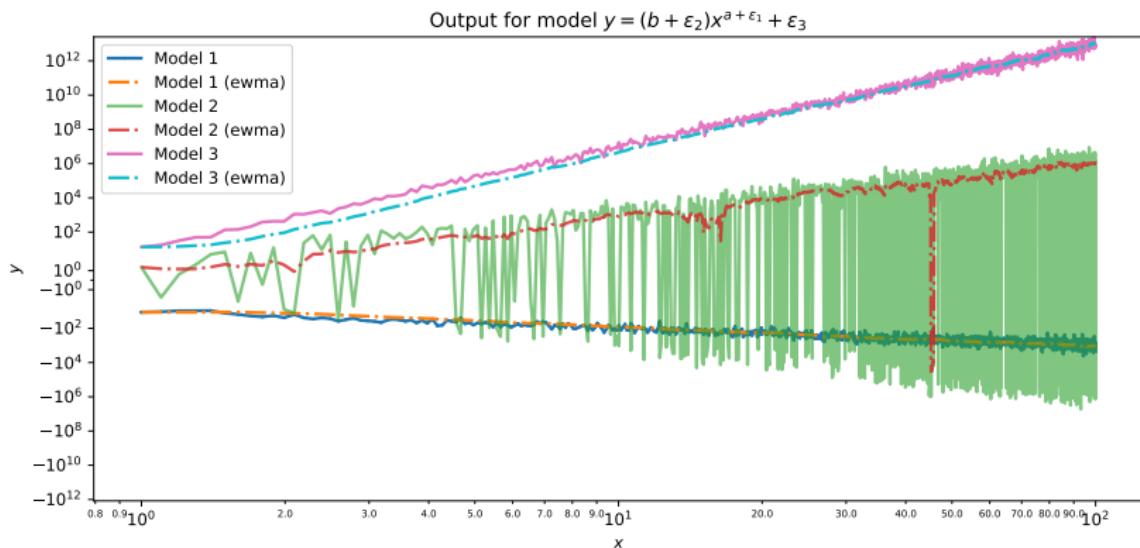
# Используйте векторную графику

- ▶ Все графики должны быть отрисованы в векторном формате



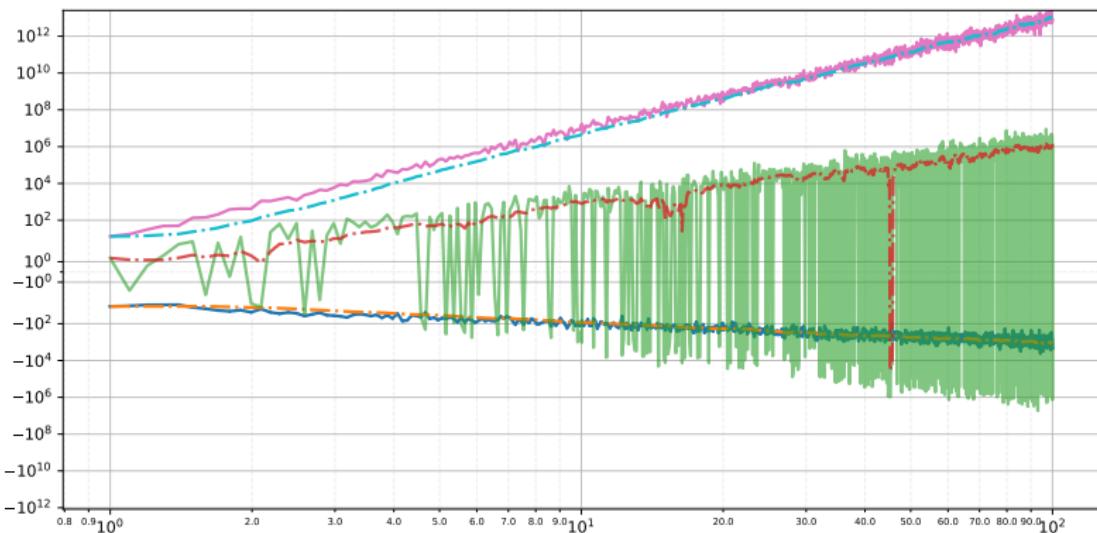
# Рисуйте сетку на графиках

- На всех графиках без исключения должна быть нарисована сетка



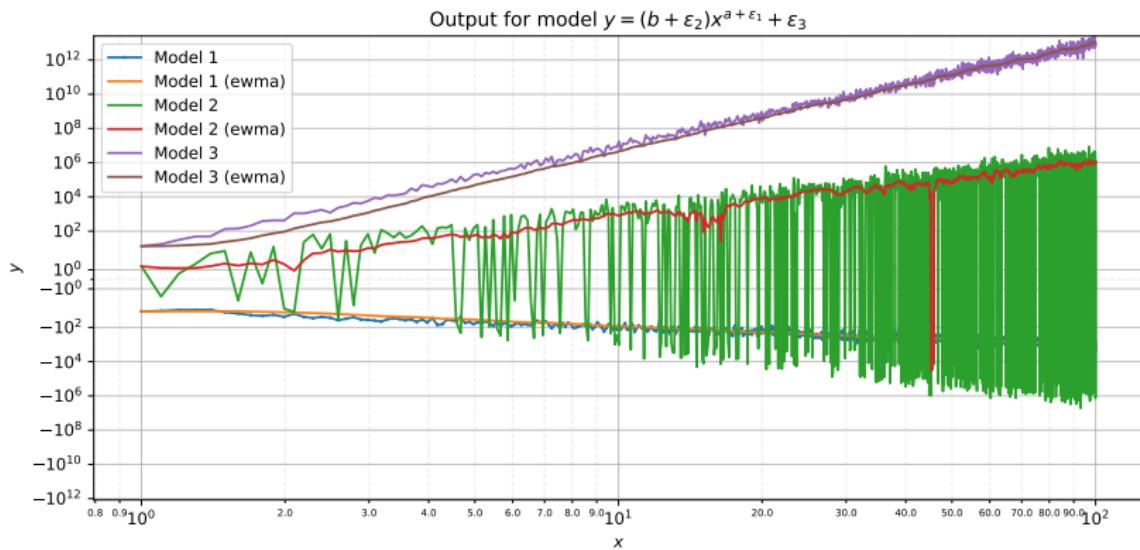
## Подписывайте все части графика

- ▶ Все графики и группы графиков должны иметь заголовок
- ▶ При необходимости оси должны быть подписаны
- ▶ Если на графике отображено несколько сущностей, то необходима исчерпывающая легенда



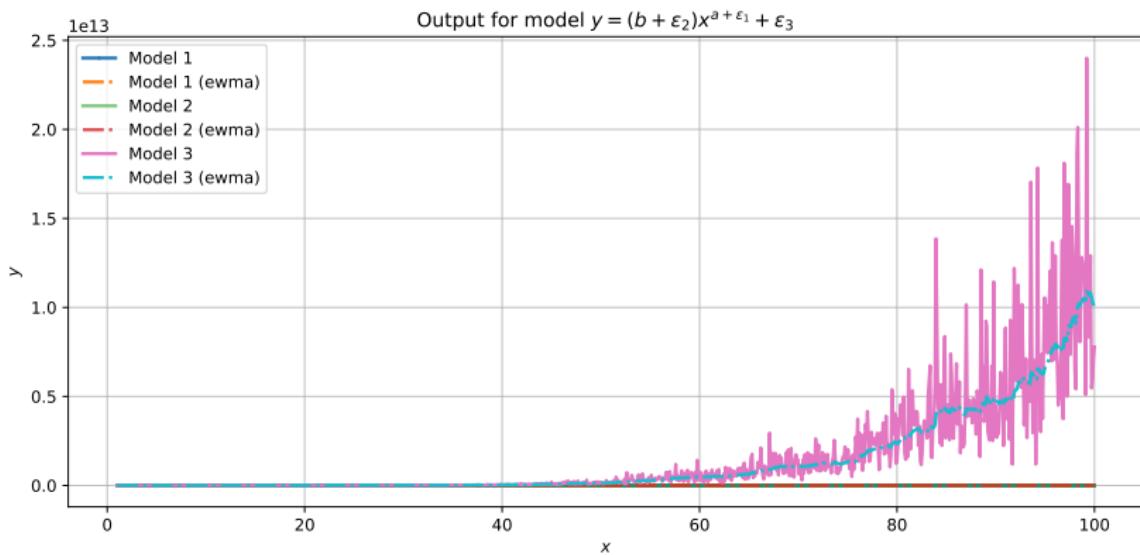
# Выбирайте правильные стили

- ▶ Все линии на графиках должны быть чётко видны
- ▶ Используйте красивую цветовую палитру с хорошо различимыми цветами



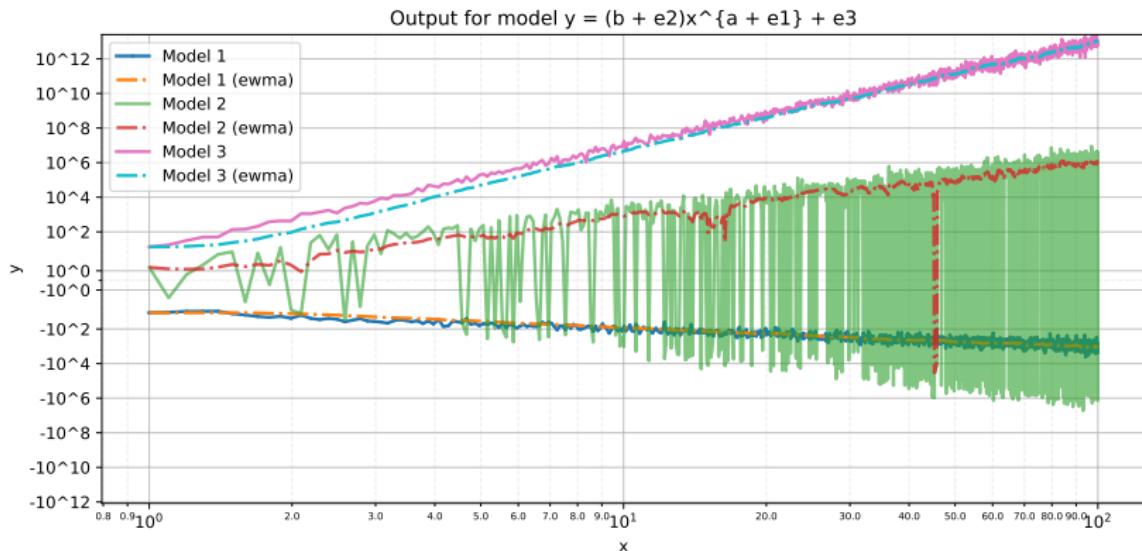
# Используйте правильный масштаб

- Масштаб по каждой оси на графике должен быть выбран правильно



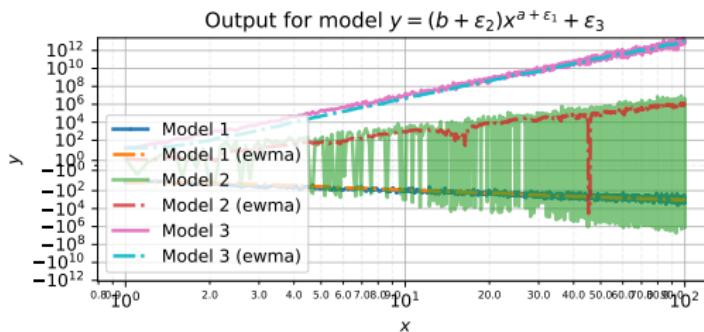
# Используйте LATEX

- Написания формул в заголовках, легенде и в подписях осей необходимо выполнять с помощью LATEX



## Подбирайте адекватный размер графика

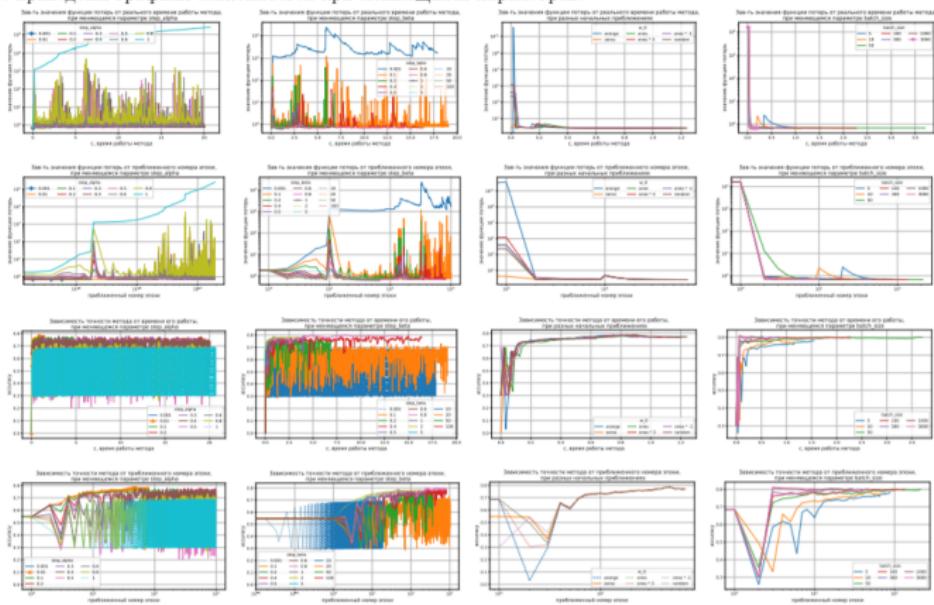
- Графики должны быть не супер-микро и не супер-макро по размерам, так, чтобы можно было увидеть все, что нужно



# Пример плохого графика из жизни

## 3.2.2 Результаты

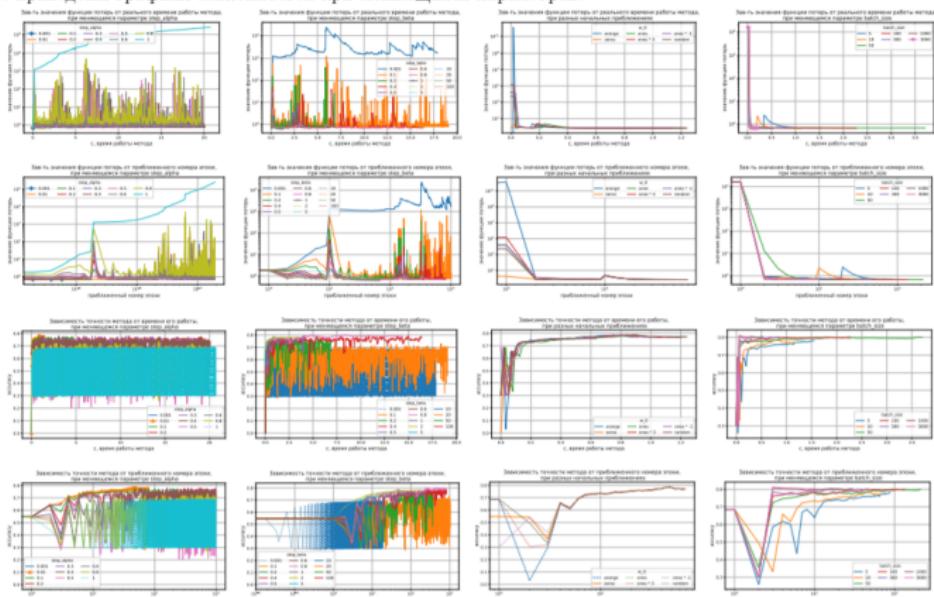
Ниже приведены графики зависимостей при меняющихся параметрах.



# Пример плохого графика из жизни

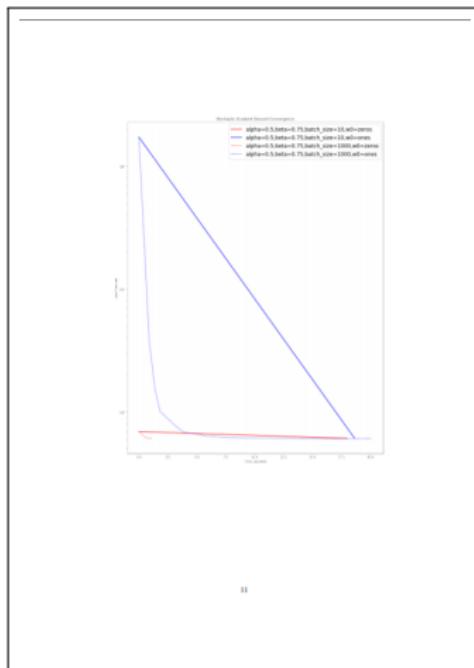
## 3.2.2 Результаты

Ниже приведены графики зависимостей при меняющихся параметрах.

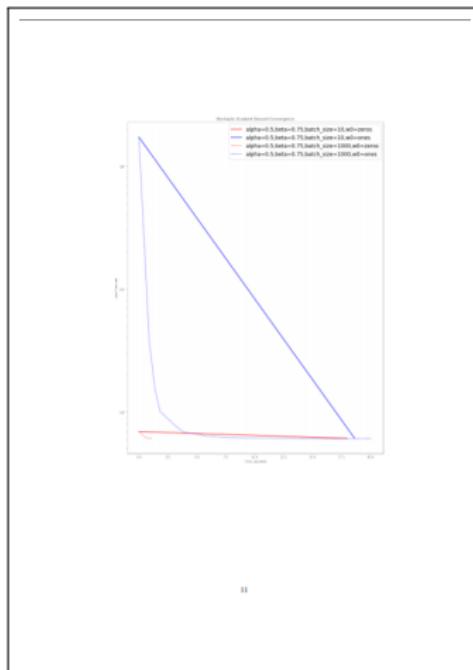


Много подобных графиков, но все они нечитаемы!

# Пример плохого графика из жизни

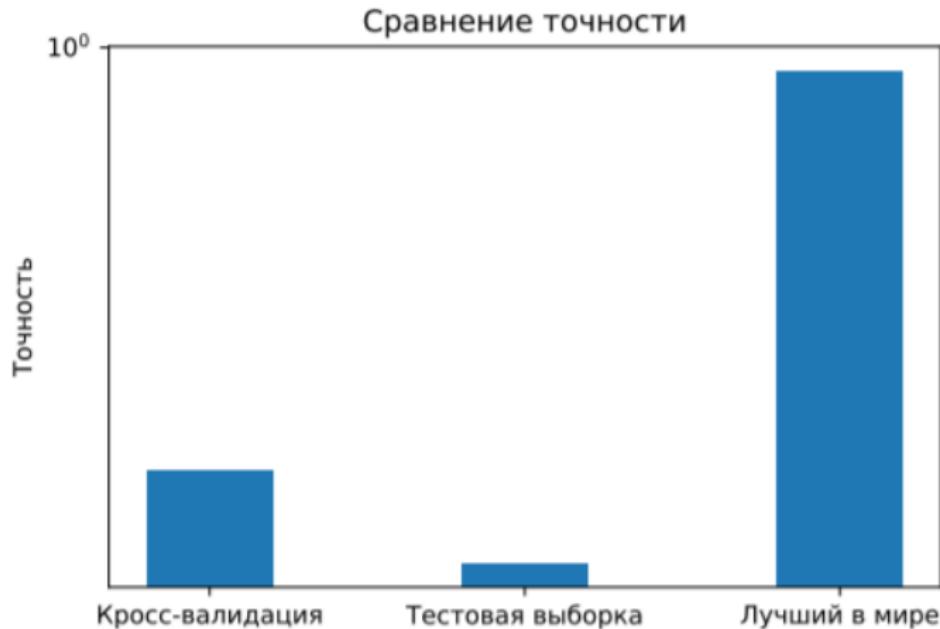


# Пример плохого графика из жизни

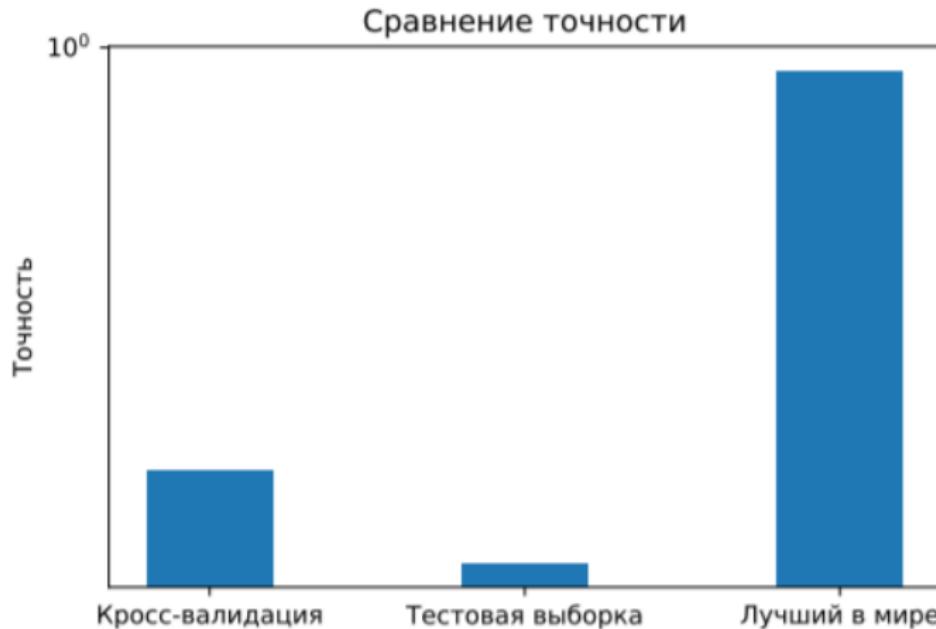


Тут — наоборот, большой график. Однако понять, что на нём происходит, нельзя.

## Пример плохого графика из жизни



## Пример плохого графика из жизни



Плохо подобраны отсчёты на вертикальной шкале.

# Пример плохого графика из жизни

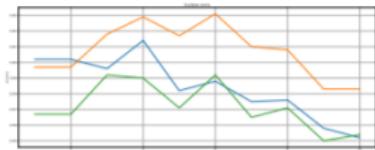


Рис. 1 Зависимость точности от числа соседей  
для евклидовой метрики

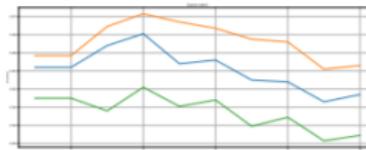


Рис. 2 Зависимость точности от числа  
соседей для косинусной метрики

# Пример плохого графика из жизни

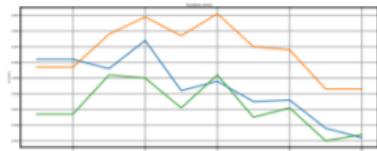


Рис. 1 Зависимость точности от числа соседей  
для евклидовой метрики

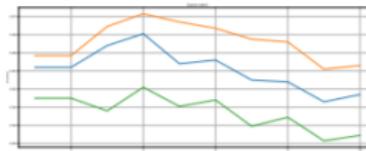
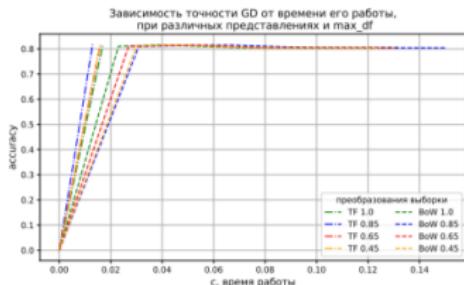


Рис. 2 Зависимость точности от числа  
соседей для косинусной метрики

Отсчёты не видно.

# Пример плохого графика из жизни

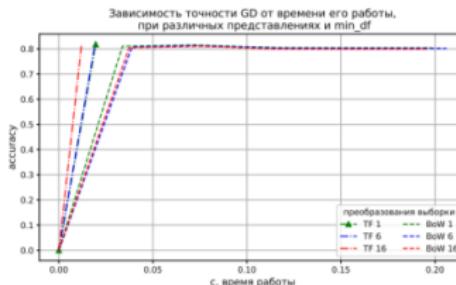
Ниже представлены графики зависимости качества от времени при различных параметрах.



Число, стоящее рядом с методом преобразования, является max\_df

Размер признакового пространства уменьшается в зависимости от увеличения min\_df: 89055

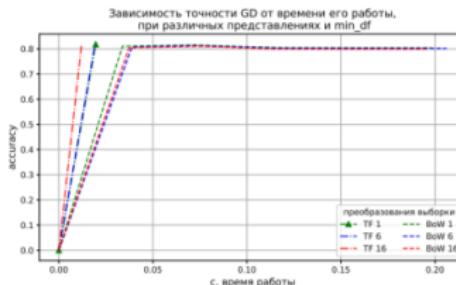
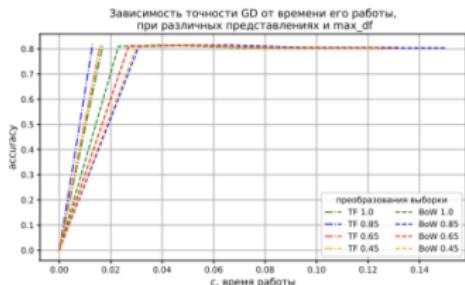
-> 15744 -> 8032. При изменении параметра min\_df размерность не меняется: 89055



Число, стоящее рядом с методом преобразования, является min\_df

# Пример плохого графика из жизни

Ниже представлены графики зависимости качества от времени при различных параметрах.



Число, стоящее рядом с методом преобразования, является max\_df

Число, стоящее рядом с методом преобразования, является min\_df

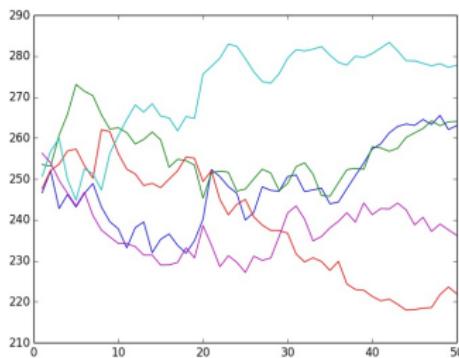
Размер признакового пространства уменьшается в зависимости от увеличения min\_df: 89055

-> 15744 -> 8032. При изменении параметра min\_df размерность не меняется: 89055

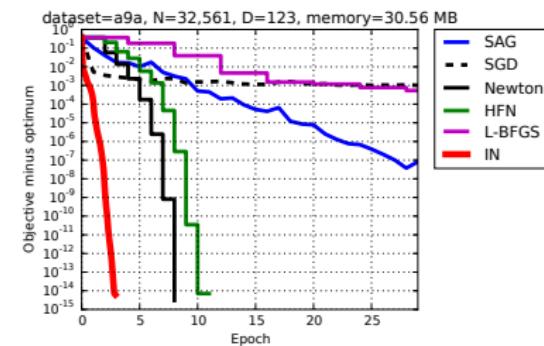
Подпись графика слишком мелкая.

# Примеры графиков из жизни

Плохой график:



Хороший график:



Элементы хорошего графика:

- ▶ Все линии жирные;
- ▶ Есть легенда;
- ▶ По осям указаны значения, сами оси подписаны;
- ▶ По осям выбрана правильная шкала;
- ▶ Сохранён в векторном формате.

# Пример хорошего графика из жизни

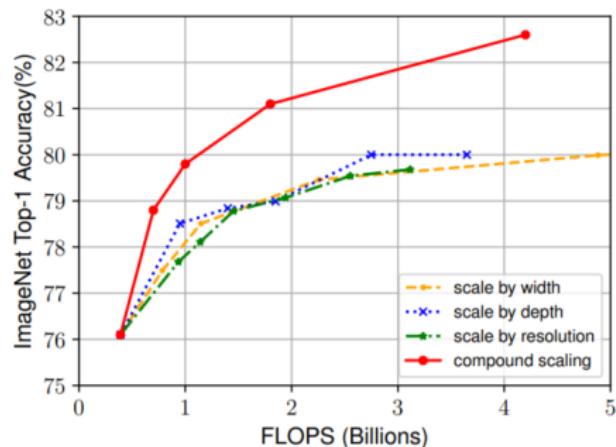


Figure 8. Scaling Up EfficientNet-B0 with Different Methods.

# Пример хорошего графика из жизни

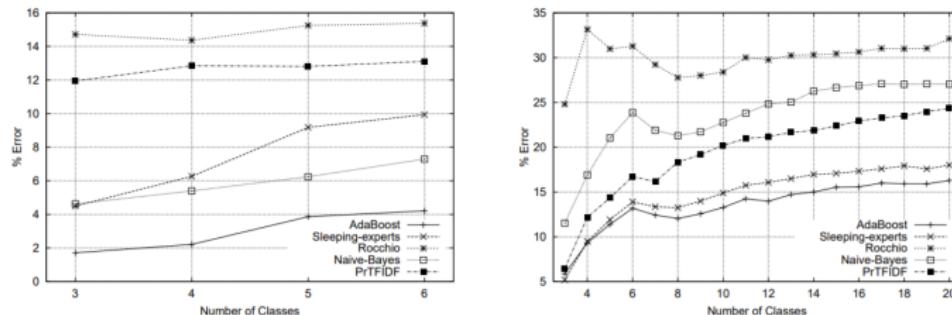
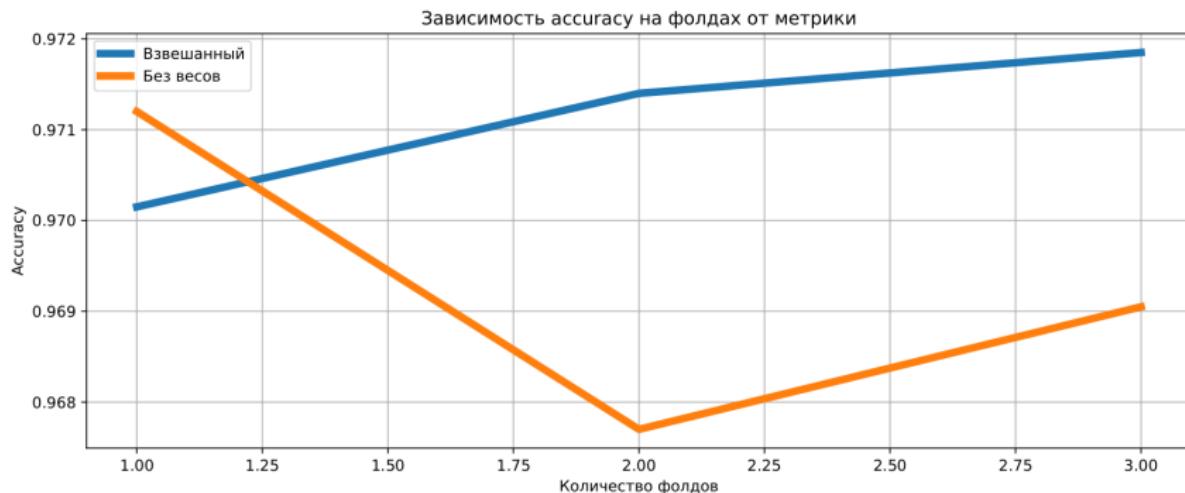


Figure 4: Comparison of error rates for AdaBoost and four other text categorization methods (naive Bayes, probabilistic TF-IDF, Rocchio and sleeping experts) as reported by Schapire and Singer [43]. The algorithms were tested on two text corpora — Reuters newswire articles (left) and AP newswire headlines (right) — and with varying numbers of class labels as indicated on the *x*-axis of each figure.

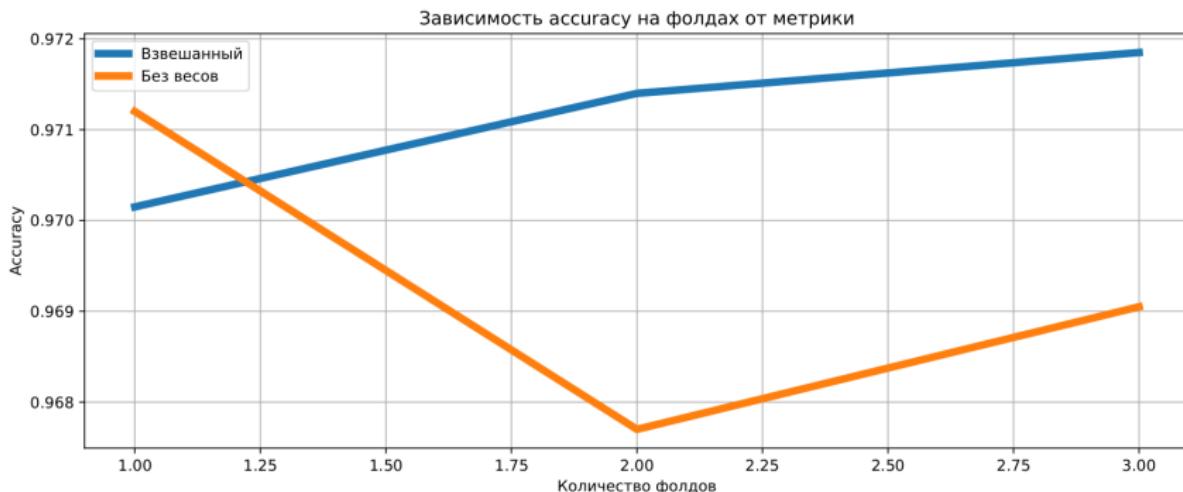
Стоит задумываться о том, как будет выглядеть график, если он будет отображаться чёрно-белым.

# Думайте об оформлении результатов



Что плохо на этой картинке?

# Думайте об оформлении результатов



Что плохо на этой картинке?

- ▶ Опечатки
- ▶ Неправильная подпись по оси  $x$
- ▶ Нет отношения порядка на оси  $x$
- ▶ Слишком подробная шкала по оси  $x$

# Пример плохо организованной страницы

## 2. Теоритическая часть

1.

$$\begin{aligned}F(w) &= \frac{1}{l} \sum_{i=1}^l \log(1 + \exp(-y_i w^T x_i)) + \lambda \frac{\|w\|^2}{2} \\ \nabla F(w) &= \frac{1}{l} \sum_{i=1}^l \nabla \mathcal{L}(M_i(w)) + \frac{\lambda}{2} \nabla \|w\|^2 \\ d\mathcal{L} &= d(\log(1 + e^{-M})) = \frac{de^{-M}}{1+e^{-M}} = \frac{-e^{-M}}{1+e^{-M}} dM = \frac{1}{e^M + 1} y_i x_i^T dw \Rightarrow \\ \nabla \mathcal{L} &= \frac{-y_i x_i}{1+e^{-M}} \Rightarrow \nabla F(w) = -\frac{1}{l} \sum_{i=1}^l \frac{y_i x_i}{1+e^{-M_i}}, \text{ где } M_i = y_i w^T x_i\end{aligned}$$

2.

$$\nabla F(X, w)_j = \frac{\partial}{\partial w_j} \left[ -\frac{1}{l} \sum_{i=1}^l \log \frac{[y_i=j] e^{(w_j, x_i)}}{\sum_{k=1}^K e^{(w_k, x_i)}} + \frac{\lambda}{2} \sum_{k=1}^K \|w_k\|^2 \right].$$

Преобразуем первую сумму:

$$\frac{\partial}{\partial w_j} \left[ \sum_{i=1}^l \langle w_j, x_i \rangle [y_i = j] - \log \sum_{k=1}^K e^{(w_j, x_i)} \right] =$$

$$\sum_{i=1}^l x_i [y_i = j] - \sum_{i=1}^l \frac{e^{(w_j, x_i)}}{\sum_{k=1}^K e^{(w_k, x_i)}} x_i \Rightarrow$$

$$\nabla F(X, w)_j = \frac{1}{l} \sum_{i=1}^l \left( \frac{e^{(w_j, x_i)}}{\sum_{k=1}^K e^{(w_k, x_i)}} - [y_i = j] \right) x_i + \lambda w_j$$

3.

$$\begin{aligned}F(X, w) &= -\frac{1}{l} \sum_{i=1}^l \left( \log P(y_i = 1 | x_i) + (\log P(y_i = -1 | x_i)) \right) = \\ &= -\frac{1}{l} \sum_{i=1}^l \left( \log \frac{[y_i=1] e^{(w_1, x_i)}}{e^{(w_1, x_i)} + e^{(w_2, x_i)}} + \log \frac{[y_i=-1] e^{(w_2, x_i)}}{e^{(w_1, x_i)} + e^{(w_2, x_i)}} \right) = \\ &= \begin{cases} w_1 = 0 \end{cases} = -\frac{1}{l} \sum_{i=1}^l \log \frac{[y_i=-1] e^{(w_2, x_i)}}{1 + e^{(w_2, x_i)}} = \\ &\frac{1}{l} \sum_{i=1}^l \log \left( 1 + [y_i = -1] e^{(w_2, x_i)} \right) = \frac{1}{l} \sum_{i=1}^l \log \left( 1 + e^{-y_i w^T x_i} \right)\end{aligned}$$

Формулы «съехали» и стали плохо читаться.

# Пример плохо организованной страницы

Это все равно, что минимизировать (попытка логарифм и затем упростить до -1)

$$\sum_{n=1}^N \sum_{k=1}^C \mathbb{I}[y_n = k] \ln p(y_n = k | x_n) =$$

$$= \sum_{n=1}^N \sum_{k=1}^C \mathbb{I}[y_n = k] \left( \ln \left( e^{w_1^T x_n} + \dots + e^{w_C^T x_n} \right) - w_k^T x_n \right) = N \cdot L_m(X, y)$$

Это и есть функция потерь для многоклассовой логистической регрессии.

Также

$$\nabla_{w_i} L_m = \frac{1}{N} \sum_{n=1}^N \sum_{k=1}^C \mathbb{I}[y_n = k] \left( \frac{e^{w_i^T x_n}}{e^{w_1^T x_n} + \dots + e^{w_C^T x_n}} - \delta_{ik} \right) x_n$$

где  $\delta_{ik} = \mathbb{I}[i = k] = k \cdot \mathbb{I}[i = k]$  - единичная Кронекера.

**Пункт 3**

Показано, что при количестве классов – 2 задача многоклассовой логистической регрессии сводится к бинарной логистической регрессии. Действительно, при  $C = 2$  и  $y \in \{-1, +1\}$  имеем

$$L_m(X, y) = \frac{1}{N} \sum_{n=1}^N (\mathbb{I}[y_n = 1] \ln \left( 1 + e^{(w_1^T x_n)^2 / w_2^T x_n} \right) + \mathbb{I}[y_n = -1] \ln \left( 1 + e^{(w_1^T x_n)^2 / w_2^T x_n} \right)) =$$

$$= \frac{1}{N} \sum_{n=1}^N \ln \left( 1 + e^{-w_1^T x_n w_2} \right)$$

где  $w = w_1 - w_2$ . Получена функция потерь для бинарной логистической регрессии.

**Эксперименты**

Тексты были предобработаны:

- Все символы, не являющиеся буквами и цифрами, заменены на пробелы;
- Все буквы приведены к нижнему регистру;

Затем при помощи sklearn.feature\_extraction.text.CountVectorizer с параметром min\_df = 0.02 текстовый признак был преобразован в несколько числовых.

# Пример плохо организованной страницы

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	636531.5	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

Есть большие пустые пространства на странице.

# Пример плохо организованной страницы

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

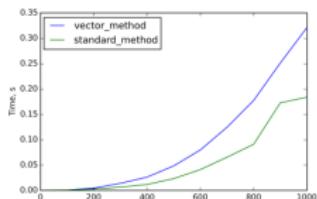


Рис. 6

Погрешность была вычислена как евклидова норма от разности двух функций:

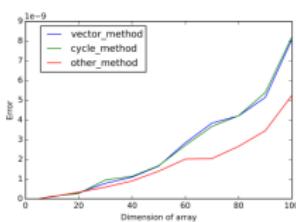


Рис. 7: Погрешность

Графики занимают слишком много места.

# Пример плохо организованной страницы

## Измерение времени

Время выполнения функций проверялось на квадратных матрицах  $X$  размера  $n \times n$  генерированных из равномерного распределения и векторах  $i, j$  размера  $n$  генерированных 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 питоне 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.

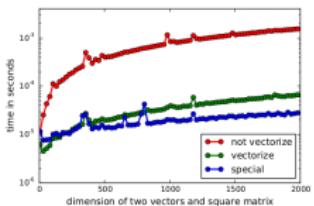


Рис. 2: Зависимость времени выполнения задачи 2 от размерности данных

Есть большие пустые пространства на странице.

# Пример плохо организованной страницы

## 10 Задача 8

### Условие

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  $(D, D)$ . Сравнить с  $scipy.stats.multivariate_normal(m, C).logpdf(X)$  как по скорости работы, так и по точности вычислений.

### Решение 1. Векторизованное

Формула плотности невырожденного нормального распределения выполнена для всей матрицы  $X$ .

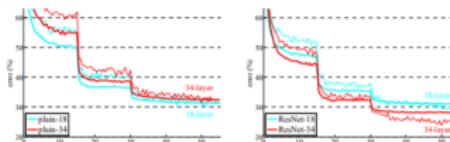
```
1 def v1_vector(X, m, C):
2     n = m.shape[0]
3     ans = -(n/2.0)*np.log(2*np.pi) - 0.5*np.linalg.slogdet(C)[1]
4     ans -= 0.5*np.dot(np.dot((X-m), np.linalg.inv(C)), (X-m).T)
5     return np.diag(ans)
```

Текст залезает на поля страницы.

# Примеры хорошо организованных страниц

Layer name	config size	18-layer	34-layer	56-layer	101-layer	152-layer
	112/112			T(7, 54, stride=2)		
				3x3 max pool, stride=2		
conv2_0	56/56	[3x3, 64] x2	[3x3, 64] x3	[3x3, 64] x3	[3x3, 128] x3	[3x3, 128] x3
conv3_1	28/28	[3x3, 128] x2	[3x3, 128] x4	[3x3, 128] x4	[3x3, 128] x4	[3x3, 128] x4
conv4_1	14/14	[3x3, 256] x2	[3x3, 256] x8	[3x3, 256] x8	[3x3, 256] x8	[3x3, 256] x8
conv5_1	7x7	[3x3, 512] x2	[3x3, 512] x3	[3x3, 512] x3	[3x3, 512] x3	[3x3, 512] x3
				[1x1, 2048]	[1x1, 2048]	[1x1, 2048]
					average pool, 1000-d, softmax	
		18-layer	34-layer	56-layer	101-layer	152-layer

Таблица 1. Архитектуры для ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3\_1, conv4\_1, and conv5\_1 with a stride of 2.



Фиг. 4. Training on ImageNet. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	<b>25.03</b>

Таблица 2. Top-1 error (%; 10-crop testing) on ImageNet validation. Here the ResNets have no extra parameter compared to their plain counterparts. Fig. 4 shows the training procedures.

34-layer plain net has higher *training* error throughout the whole training procedure, even though the solution space of the 18-layer plain network is a subspace of that of the 34-layer one.

We argue that this optimization difficulty is *unlikely* to be caused by vanishing gradients. These plain networks are trained with BN [16], which ensures forward propagated signals to have non-zero variances. We also verify that the backward propagated gradients exhibit healthy norms with BN. So neither forward nor backward signals vanish. In fact, the 34-layer plain net is still able to achieve competitive accuracy (Table 3), suggesting that the solver works to some extent. We conjecture that the deep plain nets may have exponentially low convergence rates, which impact the

reducing of the training error<sup>3</sup>. The reason for such optimization difficulties will be studied in the future.

**Residual Networks.** Next we evaluate 18-layer and 34-layer residual nets (ResNets). The baseline architectures are the same as the above plain networks, except that a residual connection is added to every pair of 3x3 filters (see Fig. 3 (right)). In the first comparison (Table 2 and Fig. 4 (right)), we use identity mapping for all shortcuts and zero-padding for increasing dimensions (option A). So there *are no extra parameters* compared to the plain counterparts.

We have three major observations from Table 2 and Fig. 4. First, the situation is reversed with residual learning – the 34-layer ResNet is better than the 18-layer ResNet (by 2.8%). More importantly, the 34-layer ResNet exhibits considerably lower training error and is generalizable to the validation data. This indicates that the degradation problem is well addressed in this setting and we manage to obtain accuracy gains from increased depth.

Second, compared to our plain counterpart, the 34-layer

<sup>3</sup>We have experimented with more training iterations (3x) and still observed the degradation problem, suggesting that this problem cannot be feasibly addressed by simply using more iterations.

## 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  $\pi_1, \pi_2, \dots, \pi_k$ . 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,  $\Delta^{k-1}$ . This allows us to define quantities such as the element-wise mean  $\bar{\pi}_{\text{gp}}[z] = [\pi_1, \dots, \pi_k]^T$  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  $\pi$ :

$$z = \text{one\_hot}\left(\arg \min_i [g_i + \log \pi_i]\right) \quad (1)$$

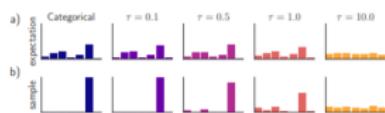
where  $g_1, \dots, g_k$  are i.i.d samples drawn from  $\text{Gumbel}(0, 1)^k$ . We use the softmax function as a continuous, differentiable approximation to arg max, and generate  $k$ -dimensional sample vectors  $y \in \Delta^{k-1}$  where

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

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

$$p_{\text{gs}, \tau}(y_1, \dots, y_k) = \Gamma(k)\tau^{k-1} \left( \sum_{i=1}^k \pi_i/y_i \right)^{-k} \prod_{i=1}^k \pi_i/(y_i)^{k+1} \quad (3)$$

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



Фиг. 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  $\tau \rightarrow 0$ . At higher temperatures, Gumbel-Softmax samples are no longer one-hot, and become uniform as  $\tau \rightarrow \infty$ .

## 2.1 GUMBEL-SOFTMAX ESTIMATOR

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

<sup>4</sup>The  $\text{Gumbel}(0, 1)$  distribution can be sampled using inverse transform sampling by drawing  $u \sim \text{Uniform}(0, 1)$  and computing  $y = -\log(-\log(u))$ .

Все числа указываются с необходимым числом знаков

угол (градусы)	fold-1	fold-2	fold-3
0	0.97475	0.9734	0.97375131
5	0.9808	0.9807	0.98160092
10	<b>0.9809</b>	<b>0.98095</b>	<b>0.98170091</b>
15	0.97965	0.979	0.97855107

Таблица 2: accuracy для разных значений углов поворота

сдвиг (пиксели)	fold-1	fold-2	fold-3
0	0.96175191	0.95740213	0.96340183
1	<b>0.97090145</b>	<b>0.96520174</b>	<b>0.96895155</b>
2	0.96700165	0.96235188	0.96460177
3	0.96565172	0.96295185	0.96355182

Число указывается с точностью до 2 или 3 знака после запятой. 35 / 39

## Избегайте длинных предложений

Далее в опытах, для настройки нашего метода, подбора оптимальных параметров, нам придётся использовать кросс-валидацию. Но эта операция занимает очень много времени, поэтому так как все стратегии, перечисленные выше, являются точными, то есть одинаково классифицируют один объект, имея одинаковую обучающую выборку, нам нужно измерить время их работы, и определить лучшую стратегию. Результат опыта на ниже приведённом рисунке.

## Чего ещё следует избегать?

- ▶ Ненаучной лексики («результаты модели получились фиговые»)
- ▶ Орфографических ошибок (установите проверку в среде)
- ▶ Грамматических, синтаксических и других ошибок
- ▶ Повествования от первого лица единственного числа
- ▶ Обращений к читателю («вашему вниманию представлены результаты экспериментов»)
- ▶ Смешения стиля использования буквы «ё» (либо везде используете «ё», либо везде «е»)

## Итог: элементы хорошего отчёта по заданию

- ▶ Отчёт подготовлен в системе  $\text{\TeX}$
- ▶ Объём отчёта: 5–20 страниц
- ▶ Текст отчёта не повторяет полной формулировки задания
- ▶ Структура отчёта соответствует пунктам задания
- ▶ Используются векторные шрифты
- ▶ Графики оформлены надлежащим образом
- ▶ Шкала для графиков выбрана правильно
- ▶ На разных графиках результаты для одинаковых методов отображаются одним и тем же цветом

## Итог: элементы хорошего отчёта по заданию

- ▶ Между расположением графиков и местами их упоминания в тексте относительно небольшое расстояние (на той же или на соседней странице)
- ▶ На страницах не должно быть много пустого места
- ▶ В большинстве случаев графики/таблицы/псевдокоды алгоритмов не должны занимать большей части одной страницы отчёта
- ▶ Все числа в тексте/таблицах указаны с необходимым числом значащих цифр
- ▶ В большинстве случаев в отчёте не должно быть никакого кода
- ▶ Для всех экспериментов описан выбранный дизайн экспериментов, а также сделаны выводы из полученных результатов