

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ М. В. ЛОМОНОСОВА
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ

Отчет по заданию №2

**«Градиентные методы обучения линейных моделей. Применение линейных
моделей для определения токсичности комментария»**

Выполнила:
студентка 317 группы
Смирнова В. С.

Москва
2023

Содержание

1 Введение	2
2 Пояснения к задаче	2
3 Теоретическая часть	2
3.1 Линейная модель бинарной классификации	2
3.1.1 Градиент функции потерь для задачи бинарной логистической регрессии.	3
3.2 Многоклассовая (мультиномиальная) логистическая регрессия	3
3.2.1 Градиент функции потерь для задачи многоклассовой (мультиномиальной) логистической регрессии	3
3.3 Сведение задачи мультиномиальной логистической регрессии при количестве классов = 2 к бинарной логистической регрессии	4
4 Эксперименты	4
4.1 Предварительная обработка данных	4
4.2 Численный подсчет градиента функции потерь и вычисление по аналитической формуле	4
4.3 Исследование поведения градиентного спуска для задачи логистической регрессии	4
4.4 Исследование поведения стохастического градиентного спуска для задачи логистической регрессии	6
4.5 Сравнение поведения двух методов	8
4.6 Лемматизация	8
4.7 Алгоритмы BagOfWords и TfIdf	9
4.8 Лучший алгоритм на тестовой выборке	11
5 Заключение	12
6 Источники	12
7 Приложения	13

1 Введение

Данное практическое задание посвящено исследованию линейных моделей и градиентных методов обучения на примере обыкновенного и стохастического градиентного спуска для решения задачи определения токсичности комментариев.

Целью работы стало исследование этих методов путем сопоставления точности и времени работы в зависимости от таких параметров как темп обучения, размер подвыборки и начальное приближение. Помимо этого, были изучены алгоритмы обработки текста: лемматизация и векторизация.

2 Пояснения к задаче

Для проведения экспериментов были модули:

1. oracles.py с реализацией функций потерь и их градиентов
2. utils.py с реализацией функции численного подсчёта градиента произвольного функционала
3. optimization.py с реализацией методов обыкновенного и стохастического градиентного спуска

Для реализации функции численного подсчёта градиента произвольного функционала использовались результаты, приведенные в теоретической части. Описание реализации обыкновенного и стохастического градиентного спуска приведены в разделах 4.3, 4.4. Эксперименты проводились на датасете, содержащим комментарии из раздела обсуждений английской Википедии, который был преобразован для решения задачи бинарной классификации: является ли данный комментарий токсичным или нет.

3 Теоретическая часть

3.1 Линейная модель бинарной классификации

Пусть $X = (x_i, y_i)_{i=1}^l$ - обучающая выборка, $x_i \subset \mathbb{R}^d$, $y_i \subset Y = \{-1, 1\}$. Линейная модель классификации определяется как $a(x) = \text{sign}(\langle \omega, x_i \rangle + b)$, где $\omega \subset \mathbb{R}^d$ - вектор весов, а $b \subset \mathbb{R}$ - сдвиг. Далее предполагается, что среди признаков есть константа, а значит $a(x) = \text{sign}(\langle \omega, x_i \rangle)$.

Процесс обучения заключается в настройке вектора весов. Величина $M_i(\omega) = y_i \langle \omega, x_i \rangle$ называется отступом объекта x_i относительно алгоритма $a(x)$. Чем больше отступ, тем более точно алгоритм классифицирует объект x_i , соответственно при $M_i(\omega) < 0$ алгоритм допускает ошибку на x_i .

Так как задачей является минимизация числа ошибок классификатора, то есть

$$\sum_i I[M_i(\omega) < 0] \rightarrow \min_w$$

Введем монотонно невозрастающую функцию $L(M(\omega))$, такую что $\mathbb{I}[M(\omega) < 0] \leq L(M(\omega))$. Тогда будем настраивать вектор весов посредством оптимизации функционала (с учетом L_2 -регуляризатора)

$$Q(X, w) = \frac{1}{l} \sum_i L(M_i(\omega)) + \frac{\lambda}{2} \|\omega\|_2^2 \rightarrow \min_w$$

Для определения вероятности принадлежности объекта к классу воспользуемся функцией log-odds (logit): $\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$, где p - вероятность принадлежности объекта к положительному классу:

$$\langle \omega, x_i \rangle = \log\left(\frac{p}{1-p}\right)$$

$$e^{\langle \omega, x_i \rangle} = \frac{p}{1-p}$$

$$p = \frac{1}{1 + e^{-\langle \omega, x_i \rangle}} = \sigma(\langle \omega, x_i \rangle)$$

Функция $\sigma(z) = \frac{1}{1+e^{-z}}$ - сигмоида. Посредством метода максимального правдоподобия можно оценить насколько вероятно получить данные значения целевой переменной при имеющейся выборке и весах, и в нашем случае имеем:

$$L(M_i(\omega)) = \log(1 + e^{-y_i \langle \omega, x_i \rangle})$$

3.1.1 Градиент функции потерь для задачи бинарной логистической регрессии.

Вычислим $\nabla_w Q(\omega, X)$:

$$\begin{aligned} \frac{\partial Q(\omega, X)}{\partial w} &= d\left(\frac{1}{l} \sum_{i=1}^l \log(1 + e^{-y_i \langle \omega, x_i \rangle}) + \frac{\lambda}{2} \|\omega\|_2^2\right) = \frac{1}{l} \sum_{i=1}^l d(\log(1 + e^{-y_i \langle \omega, x_i \rangle})) + \frac{\lambda}{2} d(\|\omega\|_2^2) = \\ &= \frac{1}{l} \sum_{i=1}^l \frac{d(e^{-y_i \langle \omega, x_i \rangle})}{1 + e^{-y_i \langle \omega, x_i \rangle}} + \frac{\lambda}{2} d(\langle \omega, \omega \rangle) = \frac{1}{l} \sum_{i=1}^l \frac{e^{-y_i \langle \omega, x_i \rangle} d(-y_i \langle \omega, x_i \rangle)}{1 + e^{-y_i \langle \omega, x_i \rangle}} + \frac{\lambda}{2} 2\langle \omega, d\omega \rangle = \\ &= \frac{1}{l} \sum_{i=1}^l \frac{e^{-y_i \langle \omega, x_i \rangle} (-y_i) \langle x_i, d\omega \rangle}{1 + e^{-y_i \langle \omega, x_i \rangle}} + \lambda \langle \omega, d\omega \rangle = \left\langle -\frac{1}{l} \sum_{i=1}^l \frac{e^{-y_i \langle \omega, x_i \rangle} y_i x_i}{1 + e^{-y_i \langle \omega, x_i \rangle}} + \lambda \omega, d\omega \right\rangle = \\ &= \langle \nabla_w Q(\omega, X), d\omega \rangle \implies \nabla Q(\omega, X) = -\frac{1}{l} \sum_{i=1}^l \frac{e^{-y_i \langle \omega, x_i \rangle} y_i x_i}{1 + e^{-y_i \langle \omega, x_i \rangle}} + \lambda \omega \end{aligned}$$

3.2 Многоклассовая (мультиномиальная) логистическая регрессия

Пусть теперь множество $Y = \{1, \dots, K\}$. Задачу многоклассовой классификации можно свести к набору бинарных задач. Пусть построено K линейных моделей $a_1(x), \dots, a_K(x)$, где $a_i(x) = \text{sign}(\langle \omega, x_i \rangle)$. Каждая модель оценивает принадлежность объекта к определенному классу. Эти оценки можно перевести в вероятности с помощью функции

$$\text{softmax}(z_1, \dots, z_k) = \left(\frac{e^{z_1}}{\sum_{k=1}^K e^{z_k}}, \dots, \frac{e^{z_K}}{\sum_{k=1}^K e^{z_k}} \right)$$

Вероятность j -го класса можно выразить так:

$$P(y = j|x) = \frac{e^{\langle \omega_j, x \rangle}}{\sum_{j=1}^K e^{\langle \omega_j, x \rangle}}$$

Обучение проводится с помощью метода максимального правдоподобия:

$$Q(X, \omega) = -\frac{1}{l} \sum_{i=1}^l P(y_i|x_i) + \frac{\lambda}{2} \sum_{k=1}^K \|\omega_k\|_2^2 \rightarrow \min_{w_1, \dots, w_k}$$

3.2.1 Градиент функции потерь для задачи многоклассовой (мультиномиальной) логистической регрессии

Вычислим $\nabla_w Q(\omega, X)$, для этого продиффицируем по z -ой компоненте вектора весов

$$\begin{aligned} \frac{\partial Q(\omega, X)}{\partial w_z} &= d_z \left(\frac{1}{l} \sum_{i=1}^l \log \left(\frac{\sum_{j=1}^K e^{\langle \omega_j, x \rangle}}{e^{\langle \omega_{y_i}, x \rangle}} \right) + \frac{\lambda}{2} \sum_{j=1}^K \|\omega_j\|_2^2 \right) = \frac{1}{l} \sum_{i=1}^l d_z \left(\log \left(\sum_{j=1}^K e^{\langle \omega_j, x \rangle - \langle \omega_{y_i}, x \rangle} \right) \right) + \\ &+ \frac{\lambda}{2} \sum_{j=1}^K d_z (\|\omega_j\|_2^2) = \frac{1}{l} \sum_{i=1}^l \frac{d_z \left(\sum_{j=1}^K e^{\langle \omega_j, x \rangle - \langle \omega_{y_i}, x \rangle} \right)}{\sum_{j=1}^K e^{\langle \omega_j, x \rangle - \langle \omega_{y_i}, x \rangle}} + \frac{\lambda}{2} \sum_{j=1}^K d_z (\langle \omega_j, \omega_j \rangle) = \\ &= \frac{1}{l} \sum_{i=1}^l \left(\frac{e^{\langle \omega_z, x \rangle} \langle x_i, d\omega_z \rangle}{\sum_{j=1}^K e^{\langle \omega_j, x \rangle}} - I[y_i = z] \langle x_i, d\omega_z \rangle \right) + \frac{\lambda}{2} 2\langle \omega_z, d\omega_z \rangle = \\ &= \left\langle \frac{1}{l} \sum_{i=1}^l \left(\frac{e^{\langle \omega_z, x \rangle} x_i}{\sum_{j=1}^K e^{\langle \omega_j, x \rangle}} - I[y_i = z] x_i \right) + \lambda \omega_z, d\omega_z \right\rangle = \langle \nabla Q(\omega, X), d\omega \rangle \\ &\implies \nabla Q(\omega, X) = \frac{1}{l} \sum_{i=1}^l \left(\frac{e^{\langle \omega_z, x \rangle}}{\sum_{j=1}^K e^{\langle \omega_j, x \rangle}} - I[y_i = z] \right) x_i + \lambda \omega_z \end{aligned}$$

3.3 Сведение задачи мультиномиальной логистической регрессии при количестве классов = 2 к бинарной логистической регрессии

Рассмотрим функцию потерь мультиномиальной логистической регрессии при $Y = \{-1, 1\}$:

$$\begin{aligned} Q(\omega, X) &= \frac{1}{l} \sum_{i=1}^l \log\left(\frac{\sum_{j=1}^K e^{\langle \omega_j, x_i \rangle}}{e^{\langle \omega_{y_i}, x_i \rangle}}\right) + \frac{\lambda}{2} \sum_{j=1}^K \|\omega_j\|_2^2 \rightarrow \min_w \\ Q(\omega, X) &= \frac{1}{l} \sum_{i=1}^l \log\left(\frac{\sum_{j=1}^2 e^{\langle \omega_j, x_i \rangle}}{e^{\langle \omega_{y_i}, x_i \rangle}}\right) + \frac{\lambda}{2} \sum_{j=1}^2 \|\omega_j\|_2^2 = \frac{1}{l} \sum_{i=1}^l \log\left(\frac{e^{\langle \omega_{-y_i}, x_i \rangle} + e^{\langle \omega_{y_i}, x_i \rangle}}{e^{\langle \omega_{y_i}, x_i \rangle}}\right) + \frac{\lambda}{2} \|\omega\|_2^2 = \\ &= \frac{1}{l} \sum_{i=1}^l \log(1 + e^{\langle \omega_{-y_i}, x_i \rangle - \langle \omega_{y_i}, x_i \rangle}) + \frac{\lambda}{2} \|\omega\|_2^2 = \frac{1}{l} \sum_{i=1}^l \log(1 + e^{-\langle \omega_{y_i} - \omega_{-y_i}, x_i \rangle}) + \frac{\lambda}{2} \|\omega\|_2^2 = \\ &= \frac{1}{l} \sum_{i=1}^l \log(1 + e^{-y_i \langle \omega, x_i \rangle}) + \frac{\lambda}{2} \|\omega\|_2^2 \rightarrow \min_w \end{aligned}$$

Так как при $\omega_{y_i} - \omega_{-y_i} = y_i \omega$ выражение под экспонентой равняется $-y_i \cdot \langle \omega_{y_i} - \omega_{-y_i}, x_i \rangle$, и при минимизации исходной функции по $\omega_{y_i}, \omega_{-y_i}$ будет проводиться минимизация полученной функции по ω , что в точности соответствует задаче бинарной классификации.

4 Эксперименты

4.1 Предварительная обработка данных

В первую очередь была проведена предварительная обработка текстов комментариев из датасета: все символы, кроме букв и цифр, были удалены, кроме того все слова были приведены к нижнему регистру с целью избежания дублирования.

Обработанные данные исходной обучающей выборки были разбиты на обучающую и валидационную в соотношении 7:3, таким образом первая состоит из 36442 объектов, а вторая - из 15619.

Затем было произведено преобразование новых выборок в разреженную матрицу, в которой значения указывают на количество вхождений каждого слова в каждый текст, при помощи конструктора `sklearn.feature_extraction.text.CountVectorizer` с параметром `min_df=0.01` (т.е токен должен встречаться как минимум в 1% документов, чтобы быть учтенным в словаре) для сокращения размерности признакового пространства до 572 и уменьшения времени работы алгоритмов. Точность в последующих экспериментах считается на валидационной выборке, если не оговорено иного.

4.2 Численный подсчет градиента функции потерь и вычисление по аналитической формуле

Было выполнено сравнение численного подсчета градиента функции потерь из `utils.py`, реализованного по формуле:

$$[\nabla f(w)]_i \approx \frac{f(x + \varepsilon e_i) - f(x)}{\varepsilon}$$

$e_i = (0, 0, \dots, 0, 1, 0, \dots, 0)$ — базисный вектор, $\varepsilon = 10^{-8} > 0$, с вычислением по аналитической формуле на обучающей выборке, полученной в прошлом эксперименте, в качестве оценки результата использовалась норма разности между векторами, равна $5.37 * 10^{-5}$, и средняя абсолютная ошибка, получившаяся $4.1 * 10^{-8}$. Данные значения близки к нулю, а значит градиенты хорошо соответствуют друг другу. Стоит заметить, что время численного подсчета градиента(6.26 s) существенно выше, чем по аналитической формуле(53.1 ms). Время работы первого алгоритма примерно в 120 раз выше, чем второго.

4.3 Исследование поведения градиентного спуска для задачи логистической регрессии

Для минимизации функционала $Q(X, w)$ выбиралось начальное приближение для вектора весов w (по умолчанию было реализовано нулевым), затем запускался итерационный процесс, на каждом шаге которого вектор w изменялся в направлении антиградиента функционала $Q(X, w)$:

$$w^{(k+1)} = w^{(k)} - \eta_k \nabla_w Q(X, w) = w^{(k)} - \frac{1}{l} \eta_k \sum_{i=1}^l \nabla_w \mathcal{L}(M_i(w))$$

Параметр $\eta_k > 0$ — темп обучения (learning rate), который считается так:

$$\eta_k = \frac{\alpha}{k^\beta}, \quad \text{где } \alpha, \beta \text{ — заданные константы}$$

Остановка алгоритма происходила при слишком малом изменении(в нашем случае 10^{-5}) функционала или после заданного числа итераций.

В этом эксперименте зафиксированы параметры коэффициент регуляризации($\lambda_2 = 0.01$), максимальное число итераций(max_iter=100). Было проанализировано поведение градиентного спуска для задачи логистической регрессии в зависимости от следующих параметров:

1. параметр размера шага $\alpha = [0.01, 0.1, 1.0, 3.0]$
2. параметр размера шага $\beta = [0.25, 0.5, 1, 2]$
3. начального приближение: нулевое, с распределениями $U(\frac{-1}{d}, \frac{1}{d})$, $N(0, \frac{1}{d})$, где d - число признаков

Сравнение проводилось по всевозможным комбинациям этих параметров(визуализация каждой возможной комбинации находится в разделе Приложения(рис 11-14)). Для демонстрации результатов экспериментов, рассмотрим поведение градиентного спуска при равномерно распределенном начальном приближении(рис 1, 2). Такой выбор начального приближения обусловлен наибольшей скоростью сходимости метода градиентного спуска (см рис.14) и наивысшей точности при сравнении с другими приближениями.

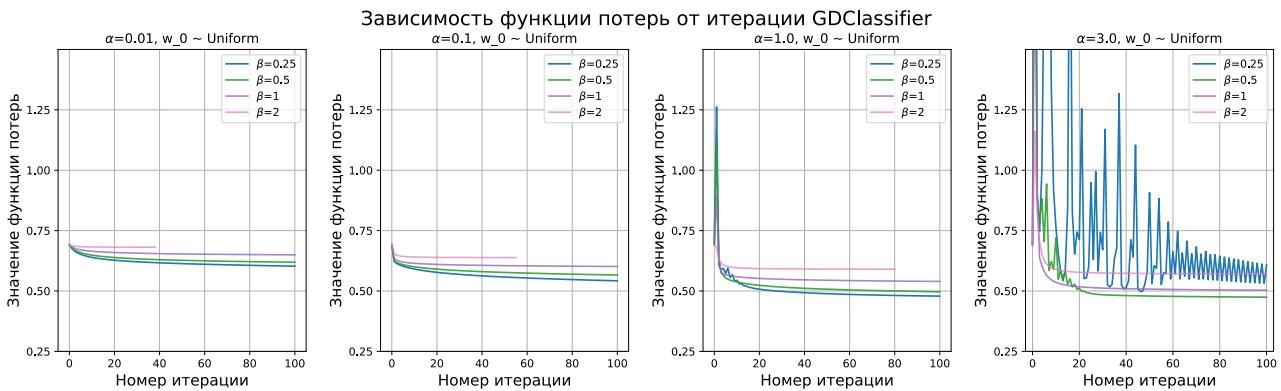


Рис. 1: График зависимости функции потерь от итерации при фиксированном начальном приближении для градиентного спуска

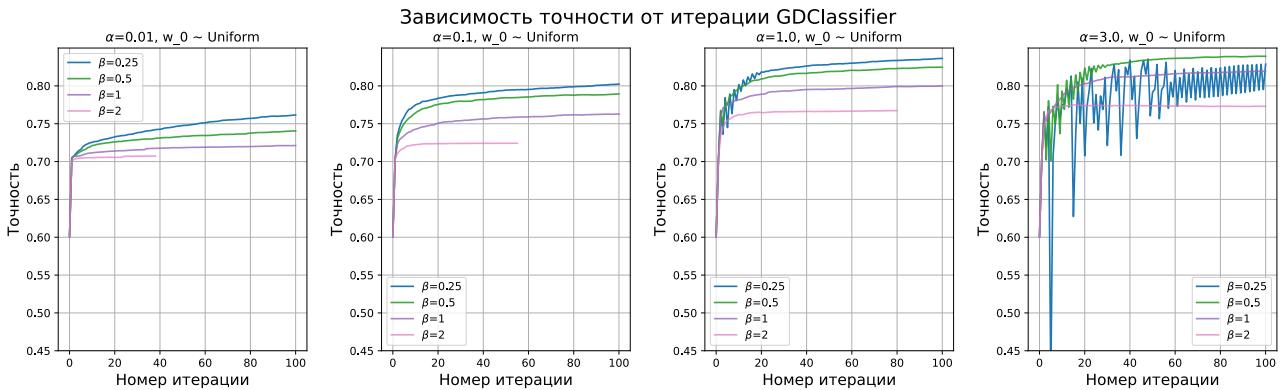


Рис. 2: График зависимости точности от итерации при фиксированном начальном приближении для градиентного спуска

Из приведенных графиков заметно, что:

- При $\alpha < 1$ функции гладкие, однако имеют среди рассмотренных комбинаций наихудшие показатели.
- При $\alpha = 1$ функции имеют заметно более лучшие показатели для любого β . Несмотря на то, что при $\beta = 0.25$, на котором достигается наилучшее значение для данного α , функции осцилируют в начале, что связано, вероятно, с большим значением темпа обучения.

- Для всех рассмотренных ранее $\alpha \leq 1$ быстрее всего метод сходится при $\beta = 2$ из-за получающегося малого η_k . Однако при данных параметрах получены наихудшие значения точности и функции потерь.
- При $\alpha = 3$ на графиках можно наблюдать большие осцилляции, радиус которых постепенно уменьшается при приближении к искомой точке экстремума функции потерь, особенно это заметно при $\beta = 0.25$. Более того, поскольку полученное при таких параметрах значение темпа обучения максимально, то, как видно из графиков, метод не сходится при 100 итерациях.
- При $\alpha = 3$ и $\beta = 0.5$ получены наилучшие показатели функции потерь и точности, несмотря на множество осцилляций в начале, примерно после 20ой итерации функция потерь имеет гладкий вид. Так как при таких параметрах метод сходится при 100 итераций, то именно они выбраны как лучшие.

Вывод: лучшие параметры модели $\alpha = 3$ и $\beta = 0.5$, начальное приближение с распределениям $U(\frac{-1}{d}, \frac{1}{d})$, где d - число признаков

4.4 Исследование поведения стохастического градиентного спуска для задачи логистической регрессии

В случае большого числа объектов вычислять на каждой итерации градиент функции потерь весьма трудоемко, вследствие чего используется стохастический градиентный спуск: необходимо оценить градиент суммы функций потерь (функционал $(Q(\omega, X))$) градиентом случайно выбранного на каждой итерации слагаемым(небольшое подмножество X размера $batch_size$). В данном случае остановка оптимизации происходила в случае, если модуль разности между соседними значениями функций был меньше точности(параметр $tolerance = 10^{-5}$). В этом эксперименте зафиксированы параметры коэффициент регуляризации($\lambda_2 = 0.01$), максимальное число эпох(max_iter=100), частота обновления(log_freq=0.5), а следующие параметры перебирались:

1. параметр размера шага $\alpha = [0.01, 0.1, 1.0, 3.0]$
2. параметр размера шага $\beta = [0.25, 0.5, 1, 2]$
3. размер подвыборки $batch_size = [500, 1000, 2500, 5000, 10\ 000]$
4. начальное приближение: нулевое, с распределениями $U(\frac{-1}{d}, \frac{1}{d})$, $N(0, \frac{1}{d})$, где d - число признаков

Для наглядности результатов эксперимент был разделен на две части:

- сначала перебирались параметры $\alpha, \beta, batch_size$ по всевозможным комбинациям, начальное приближение было выбрано нулевым по умолчанию(все результаты приведены в разделе "Приложения" 15-18)
- затем по выбранным на предыдущем этапе лучшим параметрам производился перебор начального приближения

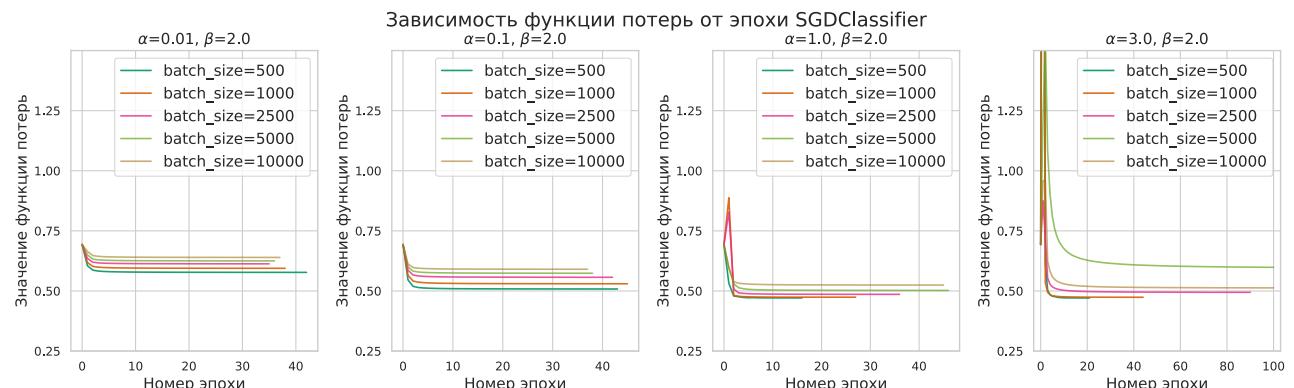


Рис. 3: График зависимости функции потерь от эпох при фиксированном начальном приближении для стохастического градиентного спуска

На рис 3-4 показаны результаты эксперимента при фиксированном $\beta = 2$, такой значение было выбрано для анализа поведения стохастического градиентного спуска, поскольку на нем достигаются наименее осциллированные графики функции потерь. Из полученных результатов можно сделать вывод, что:

- Аналогично градиентному спуску при $\alpha < 1$ функции гладкие для любого размера подвыборки, однако при таких параметрах значения функции потерь и точности наихудшие.
- При $\alpha = 1$ на графике функции потерь присутствуют резкие скачки в самом начале для параметров $batch_size = \{1000, 2500\}$, связанные вероятно с механизмом случайного выбора батча и не оптимизированных начальных весов. Стоит заметить, что на $batch_size = 500$ требуется минимальное среди всех значений эксперимента кол-во эпох для завершения работы и при этом достигаются одни из самых лучших значений точности и функции потерь.
- При $\alpha = 3$ для всех значений $batch_size$ наблюдаются резкие скачки вверх, что может быть связано с большим шагом обучения, а также с неподходящим начальным приближением, однако после первых итераций метод быстро адаптируется и значение функции потерь достигает своего минимума на $batch_size = 500$, также на этой величине достигается максимальная точность и минимальное число эпох для завершения при $\alpha = 3$.

Вывод: так как лучшие значения измеряемых величин достигается при параметрах $\alpha = 3.0, \beta = 2.0$ $batch_size = 500$ и, более того, при таких значениях метод сходится при достаточно малом числе эпох (≈ 20), то именно эти значения выбраны как лучшие для этой модели.

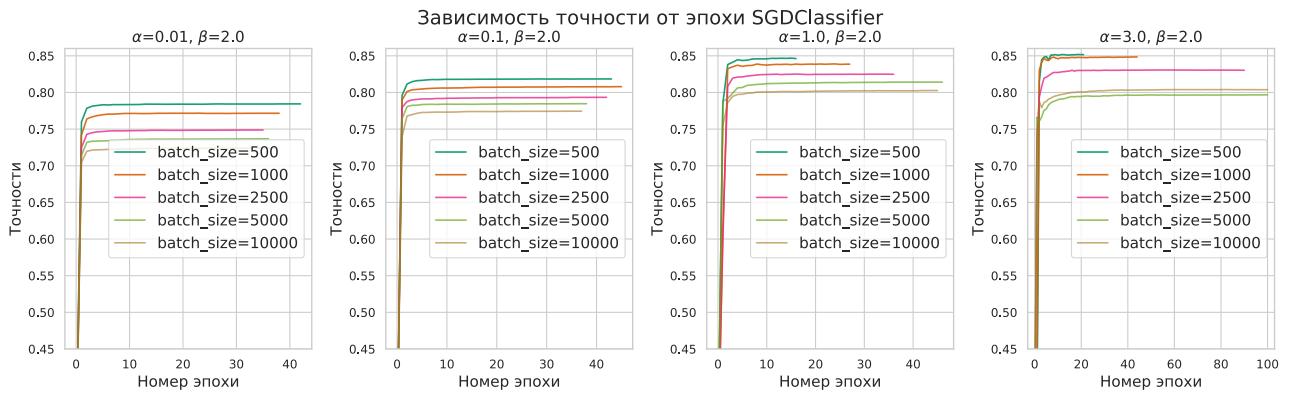


Рис. 4: График зависимости функции потерь от эпох при фиксированном начальном приближении для стохастического градиентного спуска

Для второй части эксперимента были выбраны лучшие параметры из предыдущего этапа и проанализировано влияние начального приближения на поведение модели(рис 5): При возрастании числа эпох все

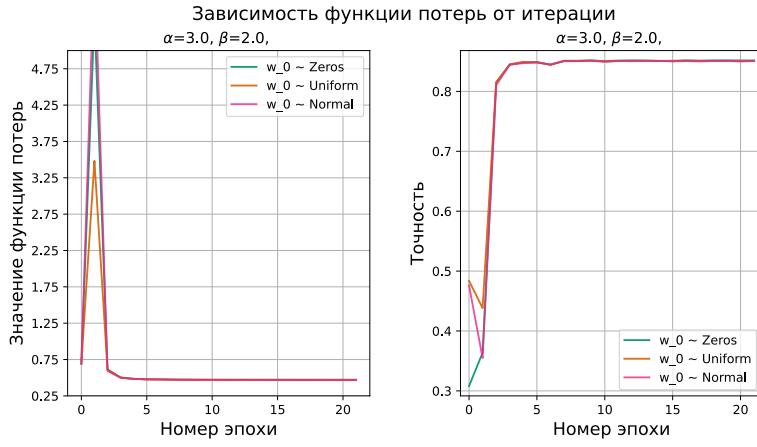


Рис. 5: График зависимости функции потерь от эпох при фиксированных $\alpha = 3.0, \beta = 2.0$ $batch_size = 500$ для стохастического градиентного спуска

начальные приближения показывают схожие значения функции потерь и точности. Однако на начальных эпохах самый лучший показатель точности и функции потерь достигается на равномерно распределенном начальном приближении, поэтому именно он выбран в качестве лучшего параметра для модели.

Вывод: лучшие параметры модели $\alpha = 3$ и $\beta = 2.0$, начальное приближение с распределением $U(\frac{-1}{d}, \frac{1}{d})$, где d - число признаков

4.5 Сравнение поведения двух методов

В этой части работы проведено сравнение обыкновенного(GD) и стохастического градиентного спуска(SGD) при лучших параметрах для каждого метода(указаны на графике), для каждого метода было выбрано начальное приближение с распределением $U(-\frac{1}{d}, \frac{1}{d})$, где d - число признаков, $batch_size = 500$. Из графиков на рис.6 можно сделать вывод о том, что *стохастический градиентный спуск показывает лучшие результаты*:

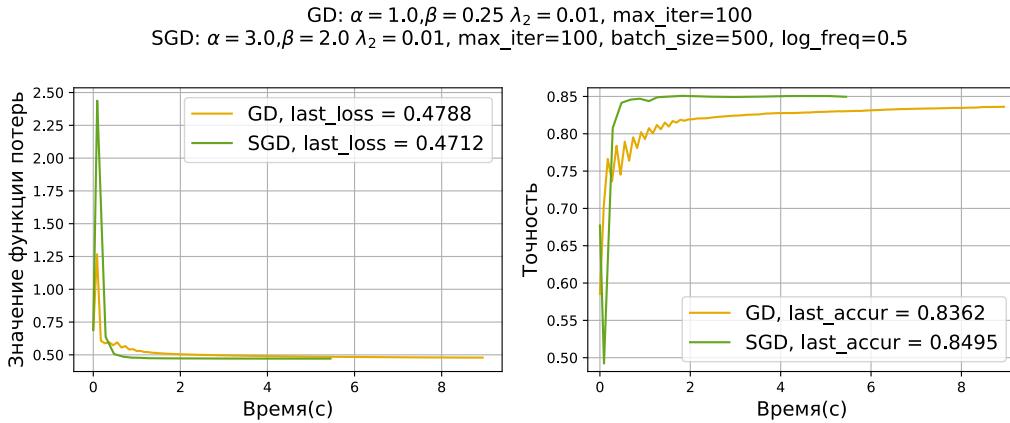


Рис. 6: График зависимости функции потерь и точности от времени для обыкновенного(GD) и стохастического градиентного спуска(SGD)

лучшие результаты: он быстрее заканчивает работу и имеет значительно большую точность, а также менее осциллированный график(только на начальном этапе имеется резкий скачок, вероятно связанный с не оптимизированными весами). В дальнейших экспериментах будут использоваться методы с лучшими параметрами, если не оговорено иного, в том числе для последних этапов работы будет рассматриваться только стохастический градиентный спуск(SGD).

4.6 Лемматизация

В данном разделе была сделана предобработка текста по следующему алгоритму:

- Приведение к нижнему регистру
- Токенизация— разделение текста на токены(элементарные единицы текста) посредством функции `nltk.tokenize.TreebankWordTokenizer()`, которая в том числе разделяет стандартные конструкции в английском языке("don't" → "do "n't")
- Лемматизация, т.е сведение разных форм одного слова к начальной форме, была выполнена с помощью функции `nltk.WordNetLemmatizer()`
- Удаление стоп-слов(предлоги, междометия, частицы) с помощью модуля `nltk.corpus`

До преобразования размер признакового пространства составлял 572, после - 452.

На рис. 7 представлена зависимость точности от времени работы методов обыкновенного(GD) и стохастического градиентного спуска(SGD) на предобработанной выборке и нет.

Был получен следующий результат: лемматизация несколько ухудшила качество(*точность GD уменьшилась на ≈ 0.002 , а SGD - на ≈ 0.01*), однако существенно сократила время работы как и обыкновенного(GD), так и стохастического(SGD) градиентного спуска(*время работы GD уменьшилось в ≈ 4 , а SGD - в ≈ 3*). А значит на больших наборах данных имеет смысл использовать лемматизацию для ускорения процесса обучения, поскольку, например, в проведенном эксперименте *размерность признакового пространства уменьшилась на 120*.

Сравнение работы методов при лемматизации и без, max_iter=100

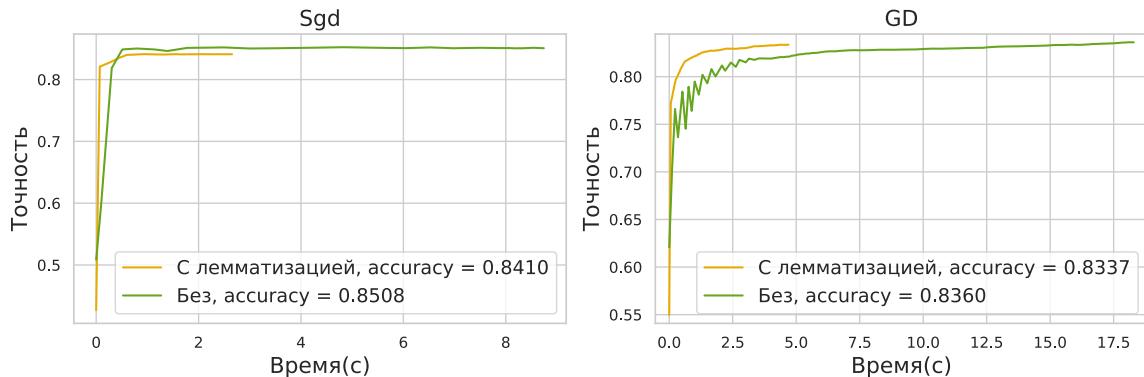


Рис. 7: График зависимости функции потерь и точности от времени для обыкновенного(GD) и стохастического градиентного спуска(SGD)

4.7 Алгоритмы BagOfWords и TfIdf

В данном разделе проведено сравнение основных подходов векторизации текста:

1. BagOfWords(BoW)

- Порядок токенов в документе не важен
- Важно сколько раз токен w входит в документ d : величина $\text{tf}(w, d)$
- Величина $\text{tf}(w, d)$ может быть введена как:

$$\text{tf}(w, d) = 1 + \log(n_{wd})$$

где n_{wd} - кол-во раз, которое токен w встретился в документе d

- Каждый документ представим в виде вектора длины $|V|$:

$$v(d) = \{\text{tf}(w_1, d), \dots, \text{tf}(w_{|V|}, d)\} \in \mathbb{R}^{|V|}$$

2. TfIdf

- В отличие от BoW учитывает распределение слов во всей коллекции текстов: величина $\text{idf}(w)$:

$$\text{idf}(w) = \log \left(\frac{|D|}{\sum_{d \in D} \mathbb{I}[w \in d]} \right) + 1$$

где D - обучающая коллекция документов

- Каждый документ представим в виде вектора длины $|V|$:

$$v(d) = \{\text{tf}(w_1, d) \cdot \text{idf}(w_1), \dots, \text{tf}(w_{|V|}, d) \cdot \text{idf}(w_{|V|})\} \in \mathbb{R}^{|V|}$$

В данном эксперименте требовалось сравнить качество, время работы алгоритма и размер признакового пространства в зависимости от следующих факторов:

- использовалось представление BagOfWords(конструктор `sklearn.feature_extraction.text.CountVectorizer()` или TfIdf(конструктор `sklearn.feature_extraction.text.TfidfVectorizer()`)
- параметров конструкторов:

1. $\text{min_df} \in \{0.0001, 0.001, 0.01\}$
2. $\text{max_df} \in \{0.35, 0.65, 0.95\}$

Такие низкие значения параметра min_df обусловлены особенностью задачи, поскольку токены- "маркеры" токсичных комментариев встречаются достаточно редко.

Значения `max_df` и `min_df`, равные x , в данном контексте означает, что токен будет исключен из рассмотрения, если он встречается в более чем $x \times 100\%$ или менее чем в $x \times 100\%$ документах, в зависимости от того, является ли это `max_df` или `min_df` соответственно. Эксперимент проводился на методе стохастического градиентного спуска (SGD) при лучших параметрах полученных в разделе 4.4 с параметром `max_iter = 300` для получения реальной точности и сходимости. Поскольку лемматизация не дала улучшения качества для SGD, то она не использовалась. Были рассмотрены всевозможные комбинации значений параметров `max_df` и `min_df`. Результаты эксперимента представлены на рис. 8-10.

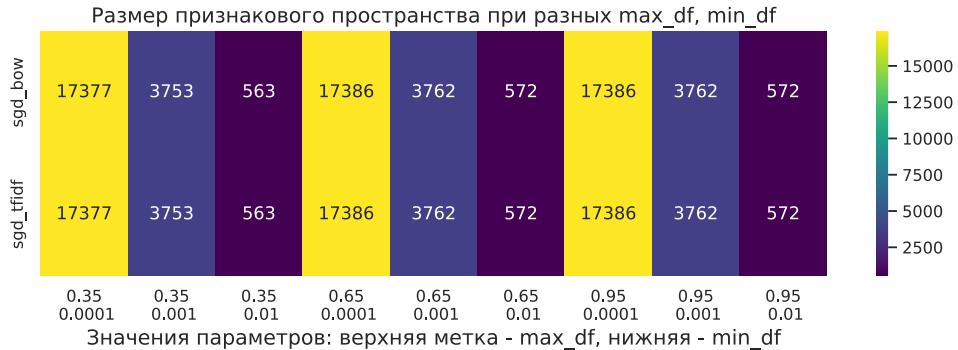


Рис. 8: Heatmap размерности признакового пространства методов Tfidf и BoW в зависимости `max_df` и `min_df` для SGD

Как можно заметить из результатов изменения количества признаков, значение `max_df` практически не повлияло на исключение токенов, так как мы получили одинаковые значения размерности для `max_df = 0.65` и `max_df = 0.95` для каждого соответствующего `min_df`. При этом параметр `max_df = 0.35` оказал влияние на размер, удалив из признаков 9 токенов, благодаря чему мы получили наивысшую точность на методе Tfidf при `min_df = 0.0001`. Также на этих признаках получено оптимальное значение времени исполнения, в связи с чем, можно назвать эту комбинацию `max_df` и `min_df` лучшей из рассмотренных.

Стоит отметить, что параметр `min_df` существенно влияет на размерность признакового пространства и, соответственно, на точность и время работы SGD, сокращая кол-во признаков в ≈ 5 и ≈ 30 для значений 0.001, 0.01 по отношению к `min_df = 0.0001`.

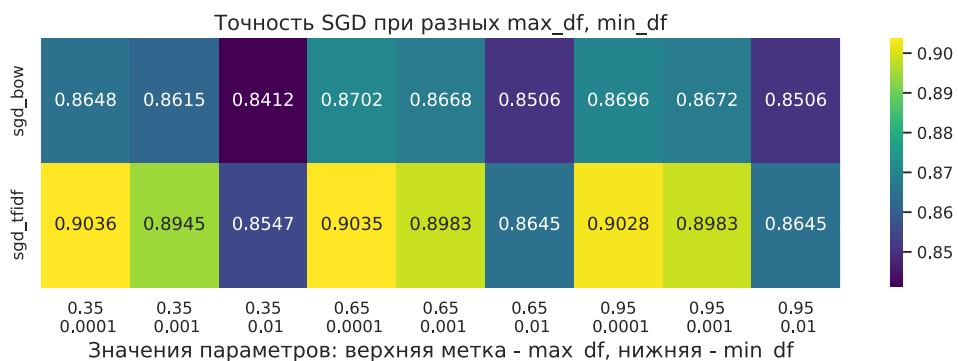


Рис. 9: Heatmap точности SGD при применении методов Tfidf и BoW в зависимости `max_df` и `min_df`

Сопоставив результаты работы методов BagOfWords и Tfidf, можно прийти к выводу, что для задачи определения токсичности комментариев больше подходит второй, так как он учитывает не только наличие слова в документе, но и его важность в контексте всего корпуса. Более того, слова, которые часто встречаются в одном документе, но редко в других, получают более высокие значения Tfidf, что важно для поставленной задачи, так как токены, определяющие токсичность комментариев используются редко(имеют неуважительный характер, что не свойственно обычному общению).

Вывод: лучшее значение точности и соответствующего времени получено при Tfidf с параметрами `max_df = 0.35` и `min_df = 0.0001`.

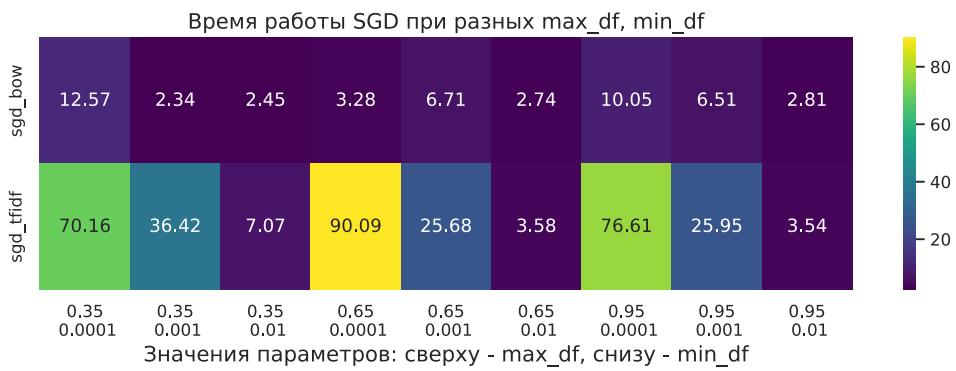


Рис. 10: Heatmap времени работы SGD при применении методов TfIdf и BoW в зависимости max_df и min_df

4.8 Лучший алгоритм на тестовой выборке

В предыдущих экспериментах точность считалась на валидационной выборке и были получены следующие параметры для лучшей модели:

- Стохастический градиентный спуск
- подобранные параметры $\alpha = 3.0$ и $\beta = 2.0$, начальное приближение с распределением $U\left(\frac{-1}{d}, \frac{1}{d}\right)$, где d - число признаков, $batch_size = 500$
- параметры по умолчанию $\lambda_2 = 0.01$, $\log_freq=0.5$, $max_iter = 10000$ (для обеспечения условия выхода по условию абсолютной разности соседних значений функции)
- Данные приводятся к нижнему регистру, а также удаляются все символы, не являющиеся цифрами и буквами, и применяется векторизация TfIdf с параметрами $max_df = 0.35$ и $min_df = 0.0001$.

Применив, данный алгоритм к тестовой выборке, получим:

- Accuracy: 0.8508
- Precision: 0.6936
- Recall: 0.9060

Полученные значения метрик свидетельствуют о том, что модель достаточно часто ложно предсказывает токсичность комментариев, тем не менее относительно общего числа объектов она достаточно точно определяет положительный класс(является ли комментарий токсичным) Рассмотрим некоторые *False-Positive объекты*

- "dear god this site is horrible" - "уверенность" в токсичность = 0.788
- "puwersa ng masa" -"уверенность" в токсичность = 0.5
- "jews are not a race because you can only get it from your mother your own mention of ethiopian jews not testing as jews proves it is not as well as the fact that we accept converts"- "уверенность" в токсичность = 0.74

На примере этих объектов, можно сделать выводы о том, что модель плохо классифицирует объекты с описками, а так же ошибается на словах, которые могут употребляться при оскорблении, хотя в данном контексте они используются либо для окраски эмоций, либо в качестве оценочных суждений. По причине наличия нецензурной лексики и других оскорблений в *False-Negative объектах* их примеры не приведены. Тем не менее, можно выделить их схожие черты:

- используется множество сокращений и сленговой лексики
- многие оскорбительные слова написаны специально неправильно с целью обхода цензуры
- содержат много терминов, связанных с названиями стран, рас, национальностей, религией

5 Заключение

Подводя итог проделанной работы, можно выделить следующие выводы:

- Стохастический градиентный спуск работает значительно быстрее обыкновенного, а так же допускает лучшие показатель функции потерь и точности
- Алгоритм лемматизации может существенно сократить время работы алгоритма
- Преобразование TfIdf с подобранными значениями max_df и min_df может существенно улучшить качество, в особенности в задачах классификации, где важны редко встречающиеся токены.

6 Источники

1. Ноутбук семинара 8 по практикуму 2022-2023
2. Конспект семинара 8 по практикуму 2022-2023

7 Приложения

Зависимость функции потерь от итерации GDClassifier

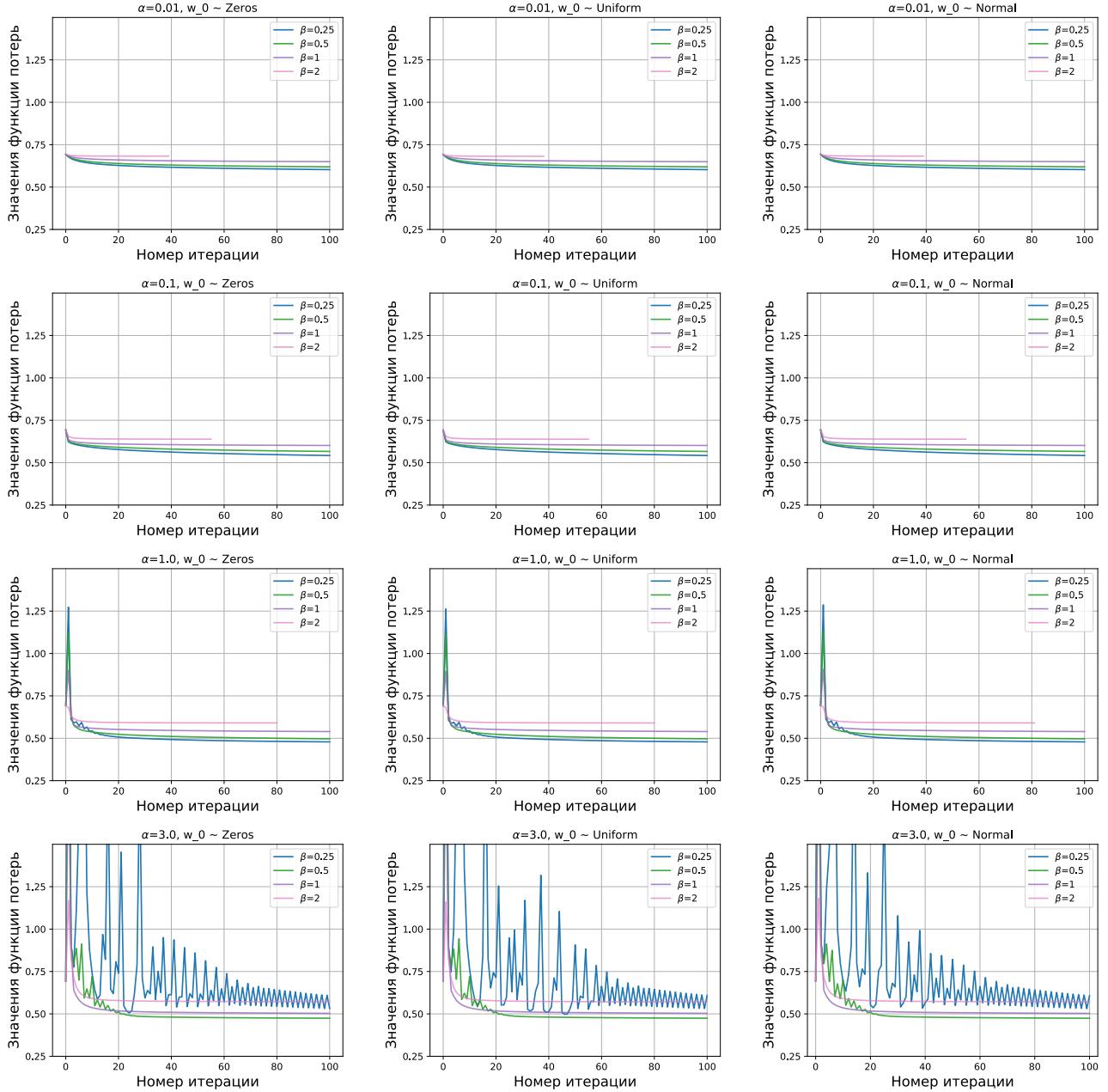


Рис. 11: График зависимости функции потерь от итерации для градиентного спуска

Зависимость функции потерь от времени GDClassifier

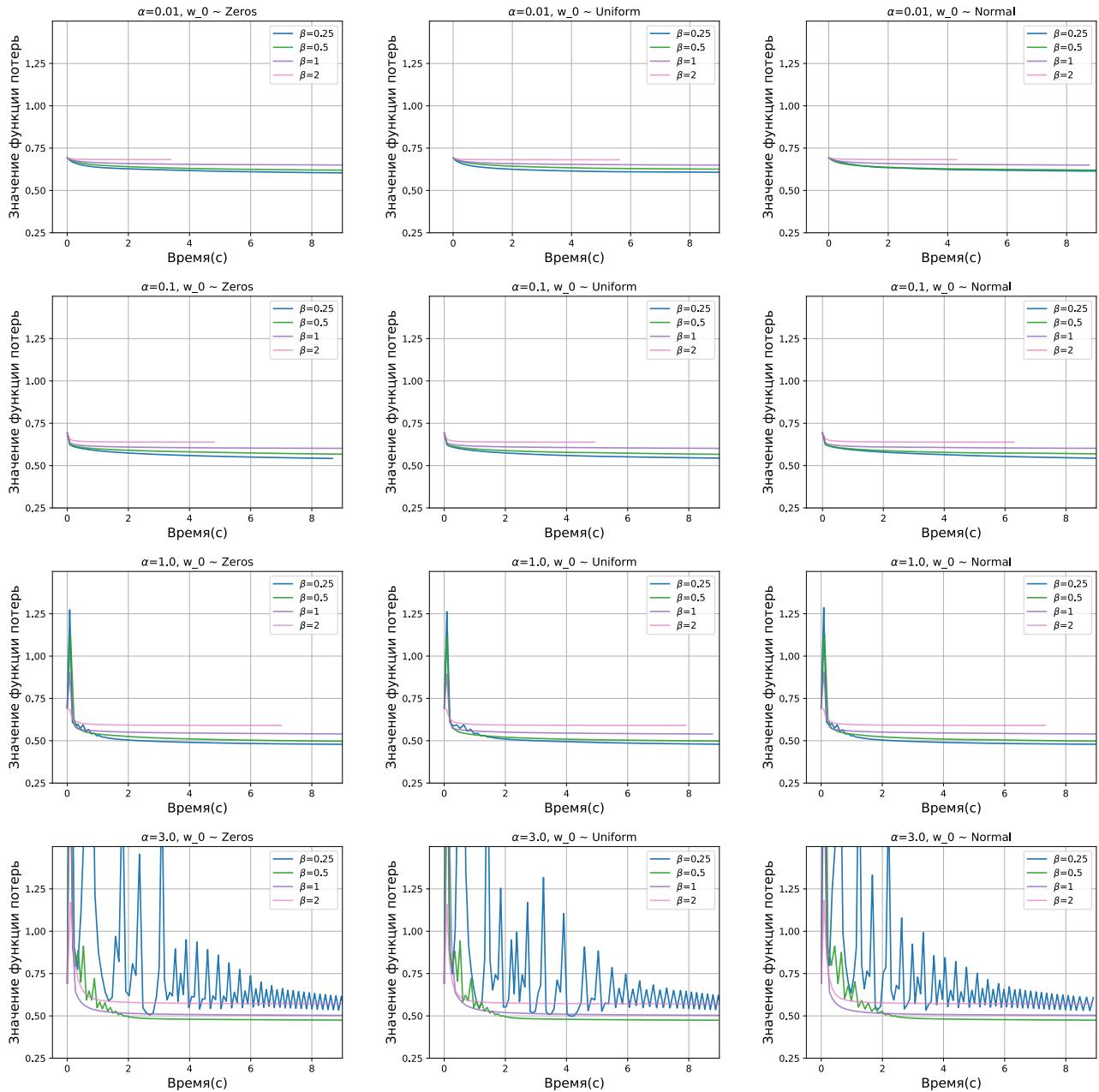


Рис. 12: График зависимости функции потерь от времени для градиентного спуска

Зависимость точности от итерации GDClassifier

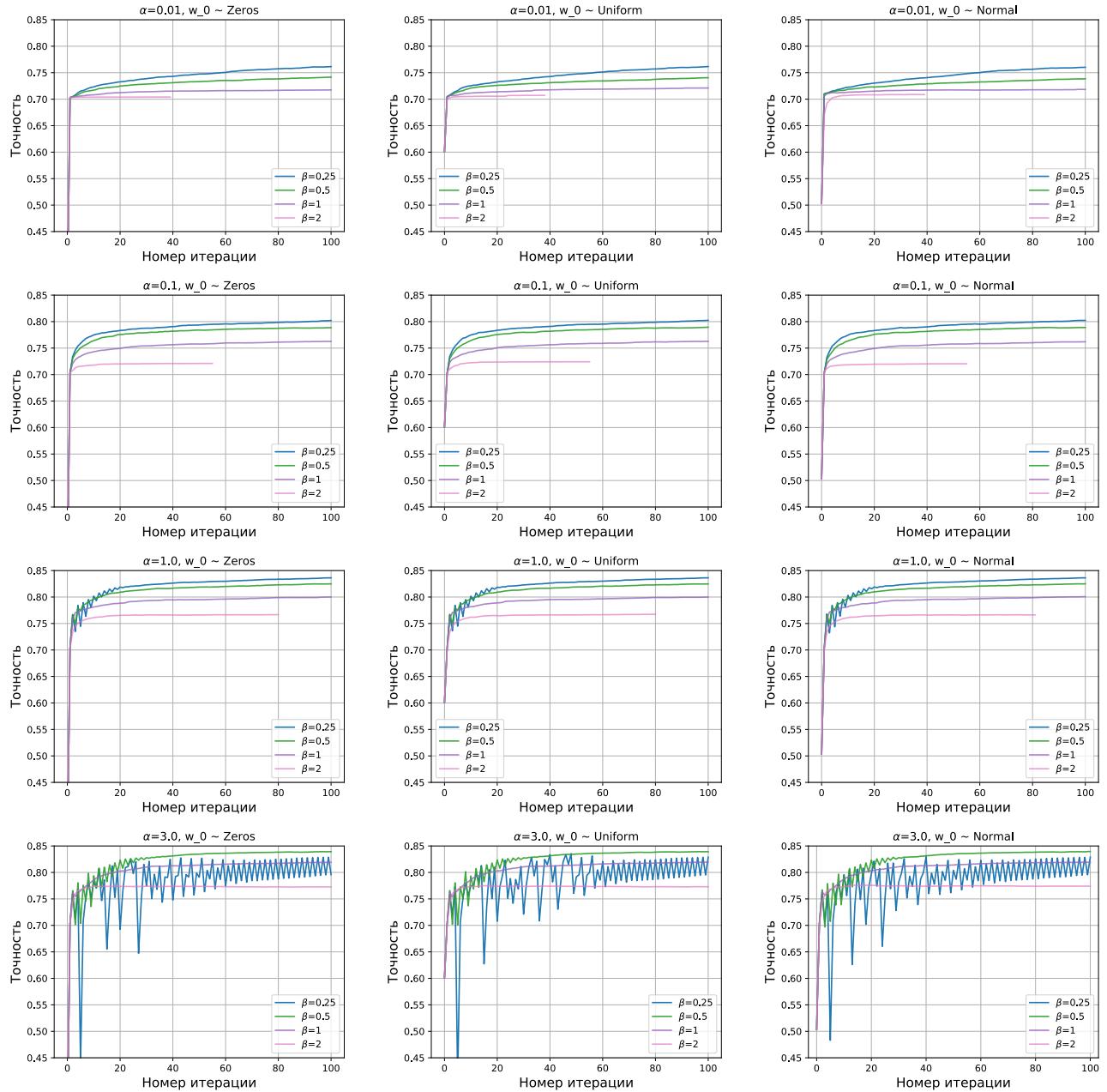


Рис. 13: График зависимости точности от итерации для градиентного спуска

Зависимость точности от времени GDClassifier

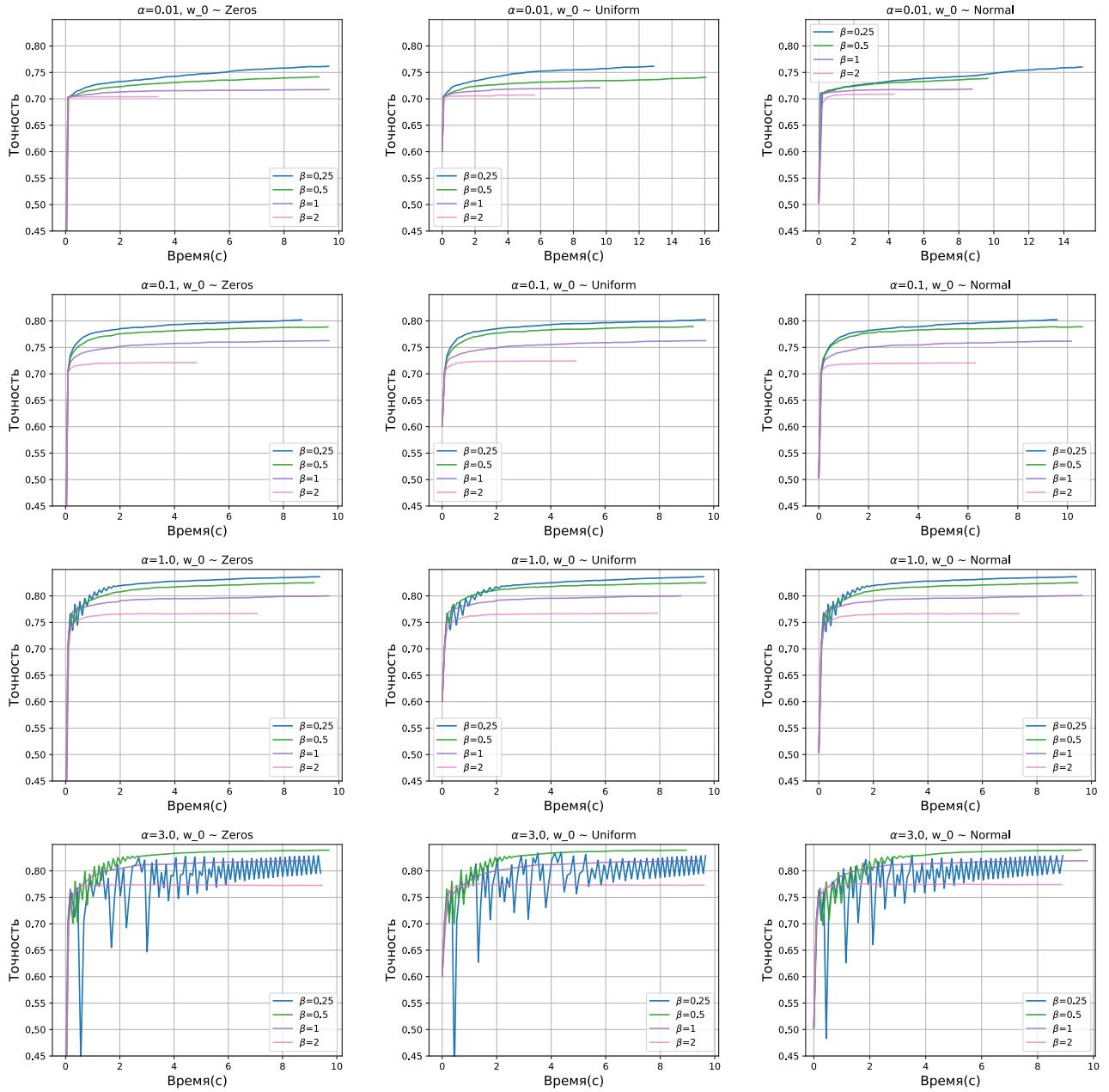


Рис. 14: График зависимости точности от времени для градиентного спуска

Зависимость функции потерь от эпохи SGDClassifier

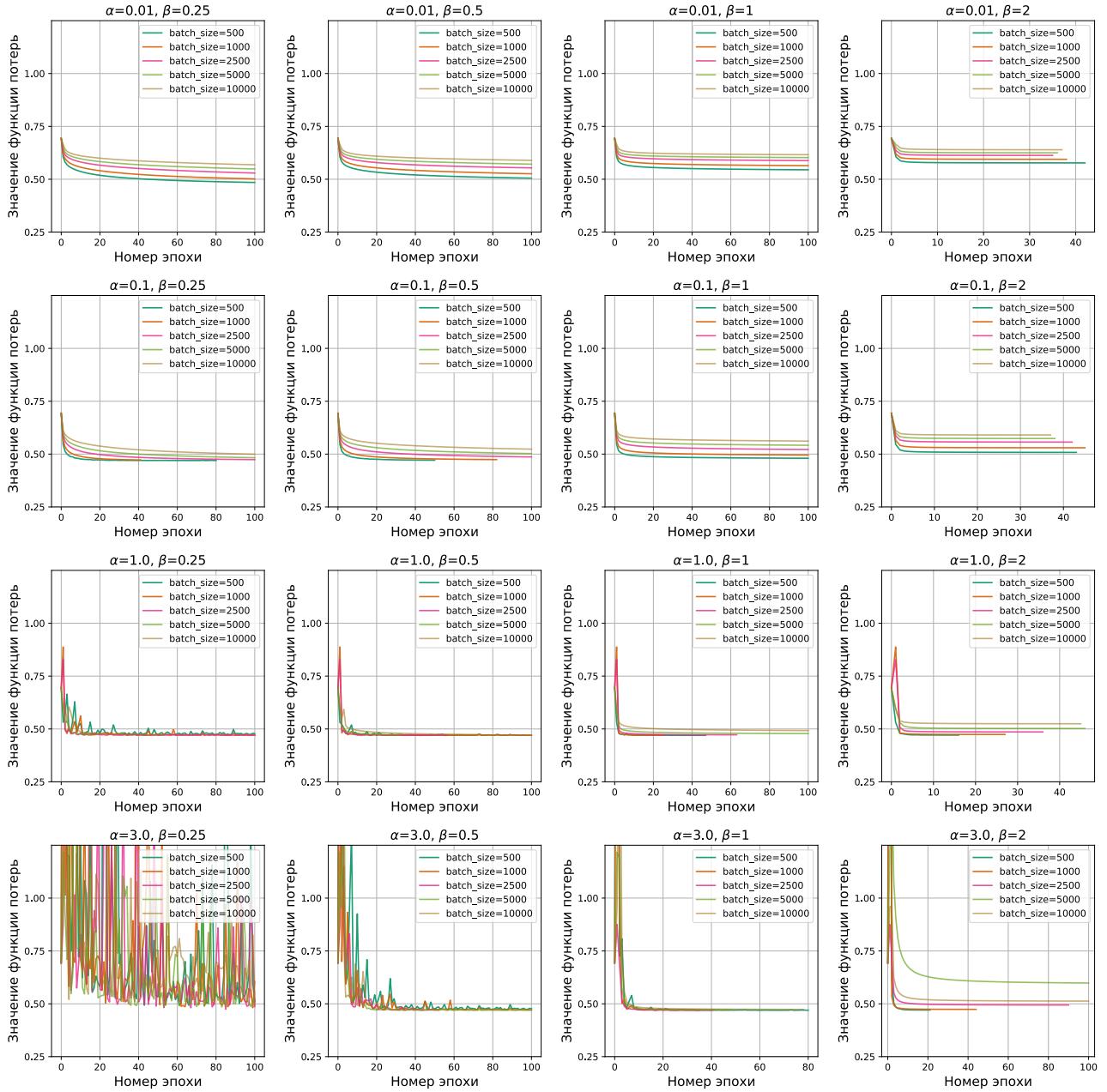


Рис. 15: График зависимости функции потерь от эпох для стохастического градиентного спуска

Зависимость функции потерь от времени SGDClassifier

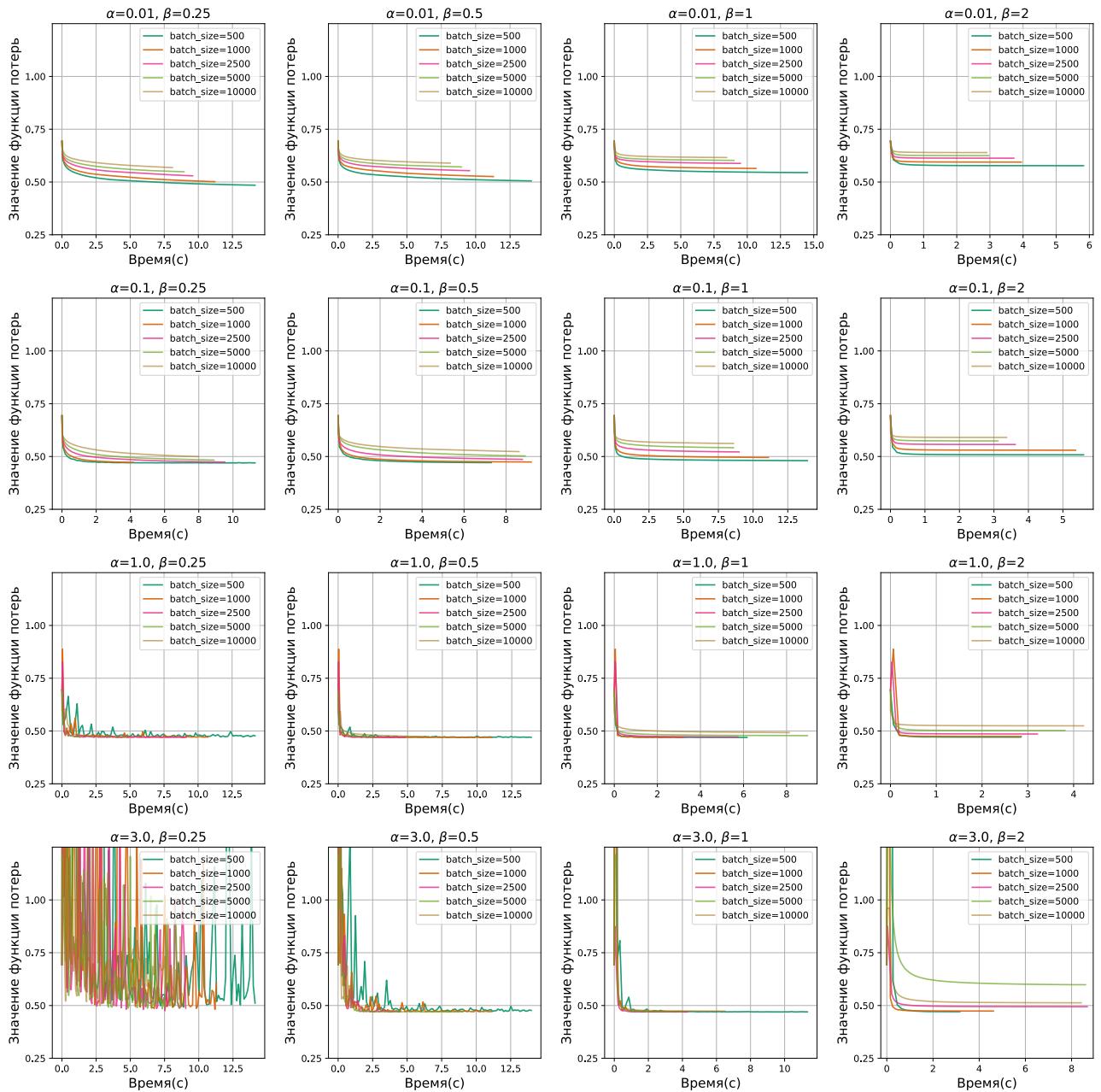


Рис. 16: График зависимости функции потерь от времени для стохастического градиентного спуска

Зависимость точности от эпохи SGDClassifier

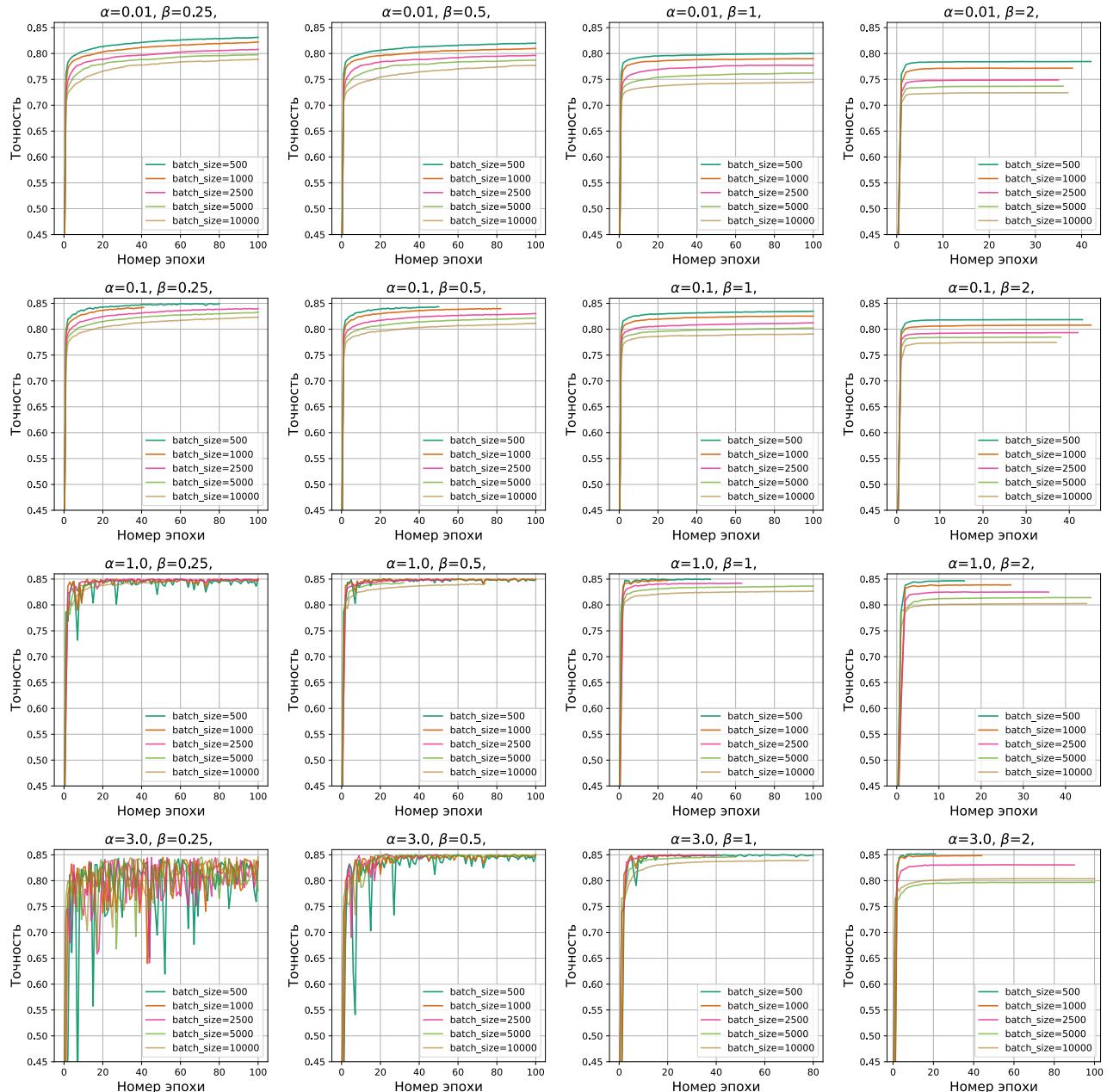


Рис. 17: График зависимости точности от эпохи для стохастического градиентного спуска

Зависимость точности от времени SGDClassifier

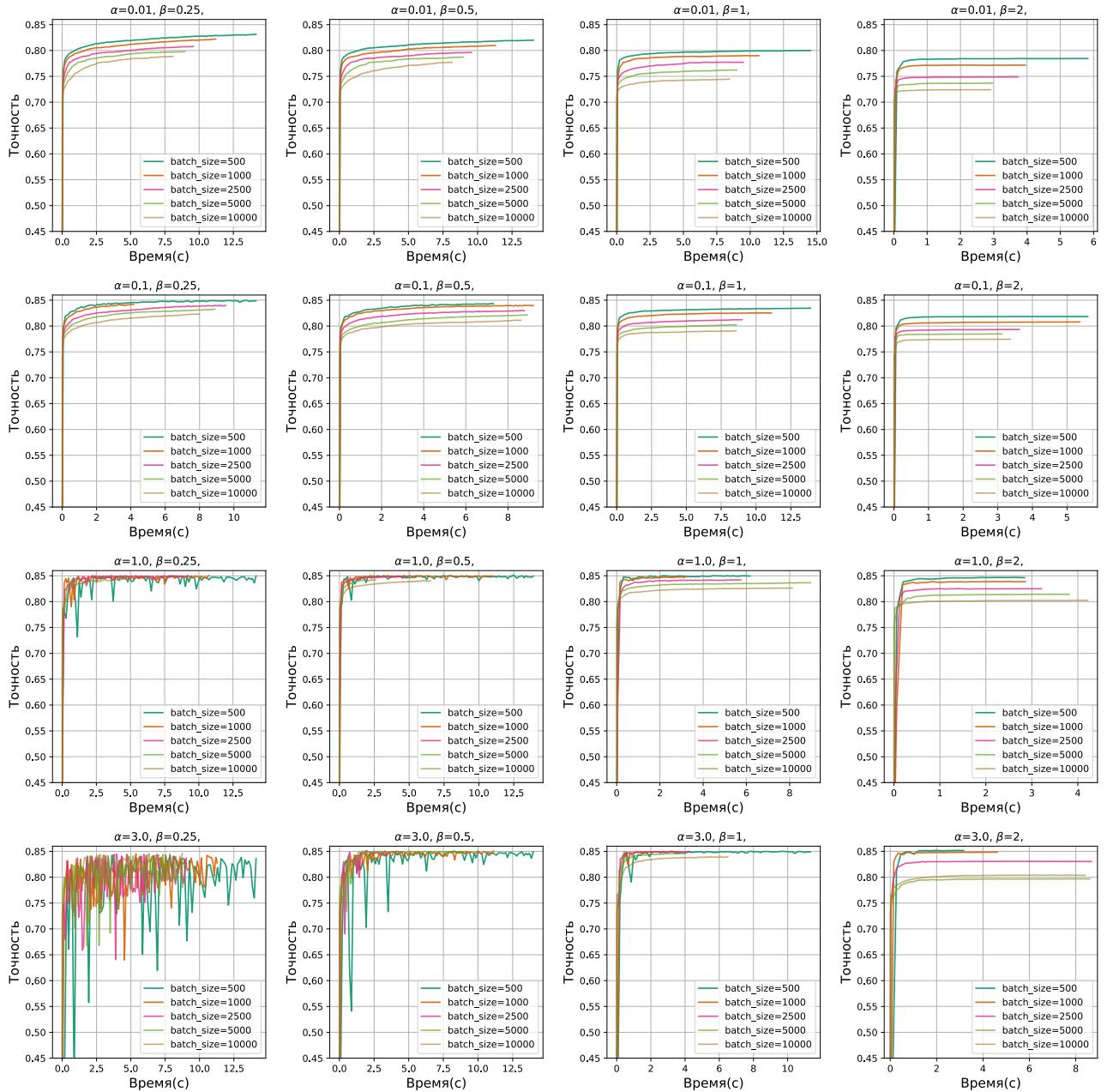


Рис. 18: График зависимости точности от времени для стохастического градиентного спуска