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
Анисимова Д. В.

Москва
2021

Содержание

1 Введение

В задании необходимо было изучить линейные модели и градиентные методы обучения, реализовать линейный классификатор. Нужно было сравнить методы градиентного спуска и стохастического градиентного спуска, исследовать их качество и скорость в зависимости от различных параметров.

2 Теоретическая часть

1. В случае бинарной логистической регрессии функция потерь имеет следующий вид:

$$Q(X, w) = \frac{1}{\ell}\sum_{i=1}^{\ell} \log P(1 + \exp(-y_i < w_i, x_i >),$$

$X \in \mathbb{R}^{\ell \times d}$ — обучающая выборка,
$y_i \in \{-1, 1\}$ — целевая переменная,
$w \in \mathbb{R}^d$ — вектор весов.
Посчитаем ее градиент по w:

$$\nabla_w Q(X, w) = -\frac{1}{\ell}\sum_{i=1}^{\ell} \frac{y_i \exp(-y_i < w, x_i >)x_i}{1 + \exp(-y_i < w_i, x_i >)}$$

2. В случае мультиномиальной логистической регрессии функция потерь выглядит так:

$$Q(X, w) = -\frac{1}{\ell}\sum_{i=1}^{\ell} \log \mathbb{P}(y_i|x_i),$$

$$\mathbb{P}(y_i|x_i) = \mathbb{P}(y = i|x_i) = \frac{\exp(< w_{y_i}, x_i >}{\sum\limits_{k=1}^{K} \exp(< w_k, x_i >)},$$

$X \in \mathbb{R}^{\ell \times d}$ — обучающая выборка,
$y_i \in \{1, \dots K\}$ — целевая переменная,
$w \in \mathbb{R}^{d \times K}$ — матрица весов.
Подставим выражение для $\mathbb{P}(y_i|x_i)$ в функцию потерь:

$$Q(X, w) = -\frac{1}{\ell}\sum_{i=1}^{\ell} \log \frac{\exp(< w_{y_i}, x_i >)}{\sum\limits_{k=1}^{K} \exp(< w_k, x_i >)} = \frac{1}{\ell}\sum_{i=1}^{\ell} \log \frac{\sum\limits_{k=1}^{K} \exp(< w_k, x_i >)}{\exp(< w_{y_i}, x_i >} =$$

$$= \frac{1}{\ell}(\sum_{i=1}^{\ell}(\log \sum_{k=1}^{K} \exp(< w_k, x_i > - < w_{y_i}, x_i >))$$

Продифференцируем эту функцию по q-й координате p-го вектора весов:

$$\frac{\partial Q}{\partial w_{pq}} = \frac{\partial}{\partial w_{pq}}(\frac{1}{\ell}\sum_{i=1}^{\ell} \log \sum_{k=1}^{K} \exp(< w_k, x_i >) - \frac{1}{\ell}\sum_{i=1}^{\ell} < w_{y_i}, x_i >) =$$

$$= \frac{1}{\ell}(\sum_{i=1}^{\ell} \frac{x_{iq} \exp(< w_p, x_i >)}{\sum\limits_{k=1}^{K} \exp(< w_k, x_i >)} - x_{iq}) = \frac{1}{\ell}(\sum_{i=1}^{\ell} \frac{x_{iq} \exp(< w_p, x_i >)}{\sum\limits_{k=1}^{K} \exp(< w_k, x_i >)} - \sum_{i=1}^{\ell} x_{iq} \mathbb{I}(y_i = p)) =$$

$$= \frac{1}{\ell} \sum_{i=1}^{\ell} (x_{iq}(\mathbb{P}(y_i|x_i) - \mathbb{I}(y_i = p))$$

Таким образом, градиент функции потерь будет представлять собой матрицу, элементами которой будут являться частные производные $\frac{\partial Q}{\partial w_{pq}}$

3. Рассмотрим функцию потерь мультиномиальной логистической регрессии в случае $\mathbb{Y} = \{-1, 1\}$:

$$Q = \frac{1}{\ell} \sum_{i=1}^{\ell} \frac{\exp(< w_{-1}, x_i >) + \exp(< w_1, x_i >)}{\exp(< w_{y_i}, x_i >} = \frac{1}{\ell} \sum_{i=1}^{\ell} (\exp(< w_{-1} - w_{y_i}, x_i >) +$$

$$+ \exp(< w_1 - w_{y_i}, x_i >)) = \frac{1}{\ell} \sum_{i=1}^{\ell} (1 + \exp(-y_i < w, x_i >))$$

3 Эксперименты

3.1 Предобработка

Были преобразованы предложения из датасета: все символы, кроме букв и цифр были удалены, а оставшиеся приведены к нижнему регистру. Приведение к нижнему регистру было необходимо, чтобы при последующем преобразовании данных в разреженную матрицу одинаковые слова, отличающиеся лишь регистром, не были приняты за разные.
Обработанные данные были преобразованы в разреженную матрицу при помощи конструктора CountVectorizer. В качестве параметра min_df было выбрано число 0.0001. Это позволило уменьшить размерность признакового пространства (она стала равной 16050) и, как следствие, немного сократить время работы алгоритма. Кроме того, как будет показано ниже, именно при таком значении min_df достигается лучшая точность алгоритма.
Обучающая выборка была разбита на обучающую и валидационную в соотношении 7:3. Точность в дальнейших экспериментах будет подсчитываться на валидационной выборке.

3.2 Исследование поведения градиентного спуска

В данном алгоритме темп обучения равен $\eta_k = \frac{\alpha}{k^{\beta+1}}$, где k — число итераций, а α и β — параметры алгоритма.
Исследуем, как зависит точность алгоритма и значение функции потерь от параметра α, положив $\beta = 0.1$, а параметр регуляризации $\lambda = 0.01$.

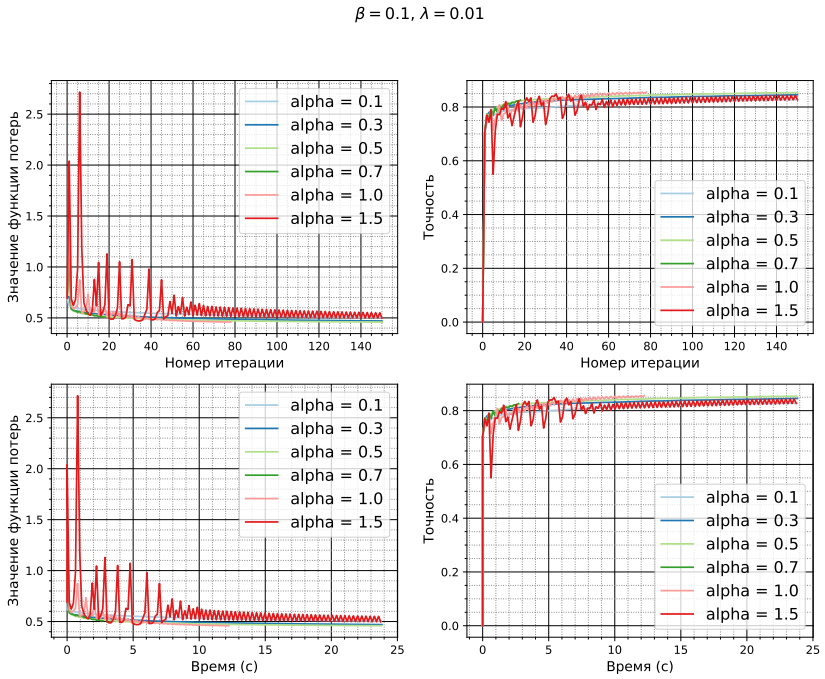


Рис. 1: Зависимость значения функции потерь и точности от времени работы и числа итераций

Отсюда видно, что при $\alpha > 1$ функции на графиках начинают очень сильно осциллировать. При таких значениях метод, если и сходится, то при большом числе итераций, что довольно затратно по времени. Такое поведение объясняется тем, что при большом значении α сильнее меняется значение функции потерь, а значит, долго не удается достичь критерия остановки.
При $\alpha < 1$ графики выглядят практически не осциллирующими, но метод показывает не самую высокую точность при не очень большом числе итераций. Кроме того, работает он долго. Это происходит потому, что с уменьшением α уменьшается и шаг, поэтому, чтобы "уйти"из окрестности начального приближения, потребуется длительное время.
Для дальнейших экспериментов будем использовать $\alpha = 1$, так как при таком значении метод показывает хорошие результаты как по точности, так и по времени работы.

Теперь будем исследовать параметр β. Параметр λ оставим равным 0.01.

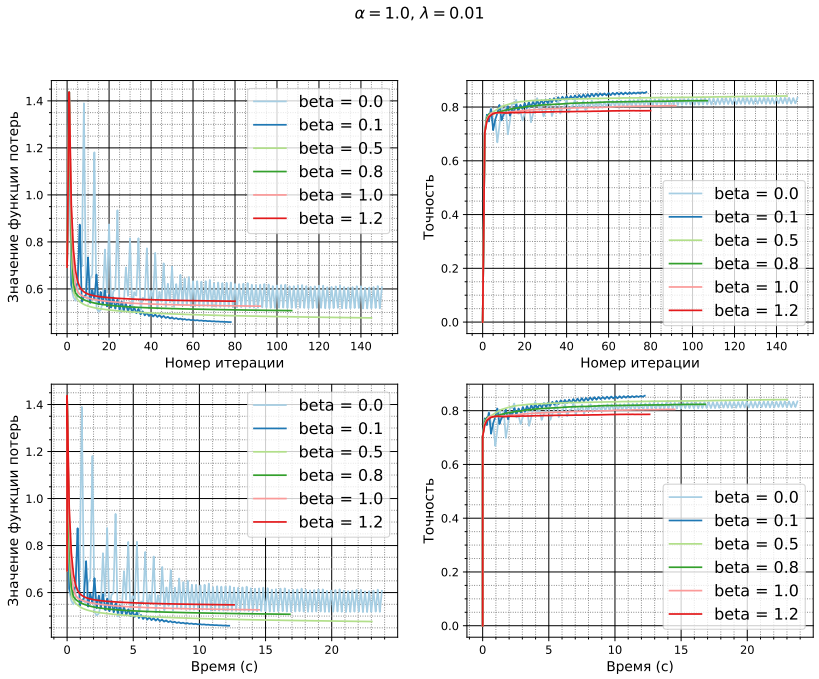


Рис. 2: Зависимость значения функции потерь и точности от времени работы и числа итераций

При $\beta = 0.0$ на графиках вновь появляются скачки, что закономерно, потому что в этом случае шаг либо равен константе. В этом случае нужно брать какое-то маленькое значение α.
При β, близких к 1 (0.8, 1.0, 1.2) осцилляций нет, но на них не достигается самая высокая точность.
Лучшая точность достигается при $\beta = 0.1$, но начальных итерациях есть скачки, поэтому при уменьшении числа итераций могут возникнуть проблемы с точностью.
При $beta = 0.5$ скачков не возникает, но увеличивается время работы. В то же время по сравнению со значениями параметра, близкими к 1, $\beta = 0.5$, показывает лучшую точность.
Для дальнейших вычислений разумным будет взять $\beta = 0.1$ или $\beta = 0.5$.

Рассмотрим параметр w_0. Сгенерируем значения этого параметра, используя нормальное, геометрическое и равномерное распределения, а также зададим нулевые и единичные веса.

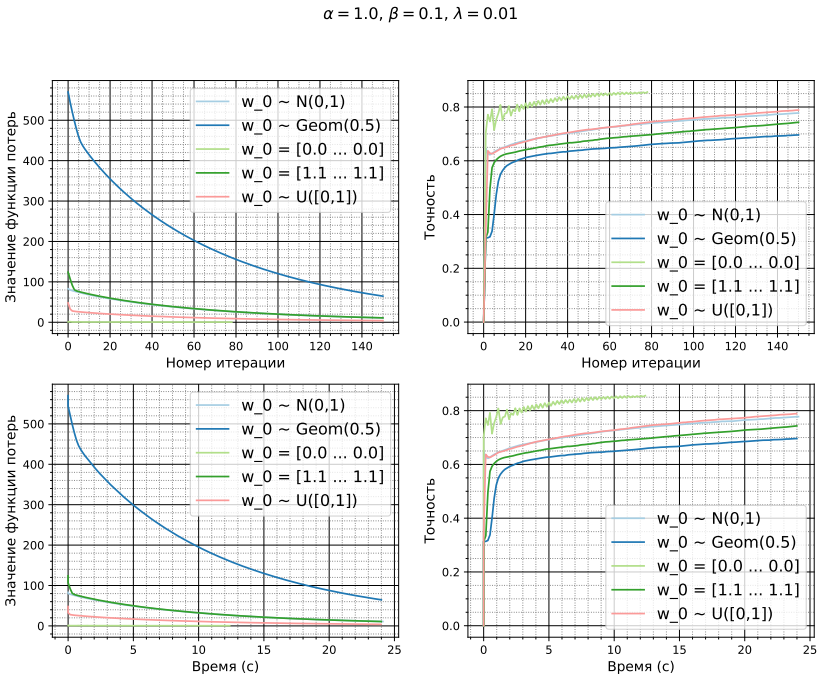


Рис. 3: Зависимость значения функции потерь и точности от времени работы и числа итераций

Как видно из графиков, наилучшим начальным приближением является нулевое. На нем достигается самая высокая точность, даже несмотря на скачки на начальных итерациях. В дальнейшем будем использовать это приближение.

3.3 Исследование поведения стохастического градиентного спуска

Точно так же, как и в градиентном спуске, шаг алгоритма равен $\eta_k = \frac{\alpha}{k^{\beta+1}}$. Исследуем все те параметры, которые были исследованы в градиентном спуске, добавив к ним размер батча.

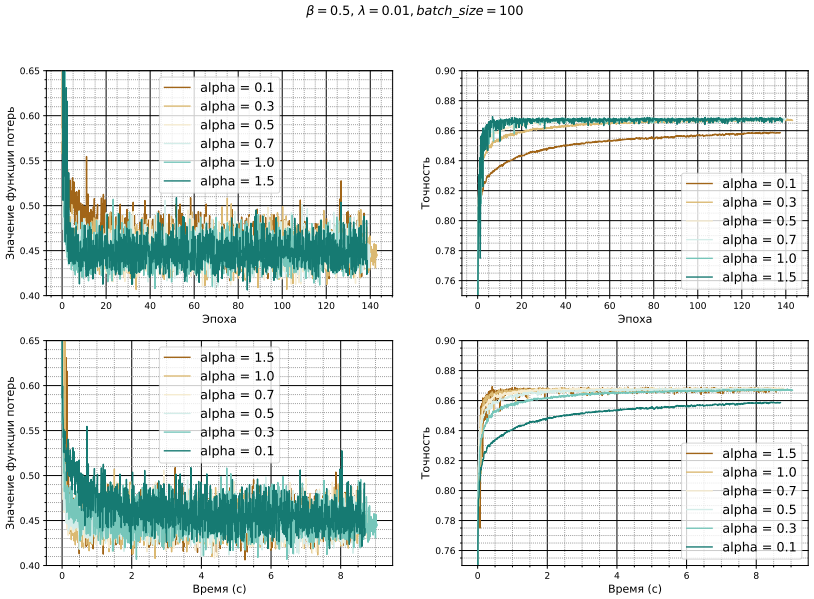


Рис. 4: Зависимость значения функции потерь и точности от времени работы и числа эпох

Точно так же, как и в градиентном спуске, здесь при больших значениях α на графике появляются очень сильные скачки. Но здесь они стали гораздо более ярко выраженными, потому что число итераций возросло по сравнению с обычным градиентным спуском.
Слишком маленькие значения α также подходят не очень хорошо, потому что на них достигается не слишком высокая точность.
Как и в случае обычного градиентного спуска, возьмем $\alpha = 1.0$, потому что при таком значении точность довольно высокая, а осцилляции не столь сильные, как при $\alpha > 1$.

Посмотрим на параметр β, взяв $\alpha = 1.0$, $\lambda = 0.01$, $batch\_size = 1000$

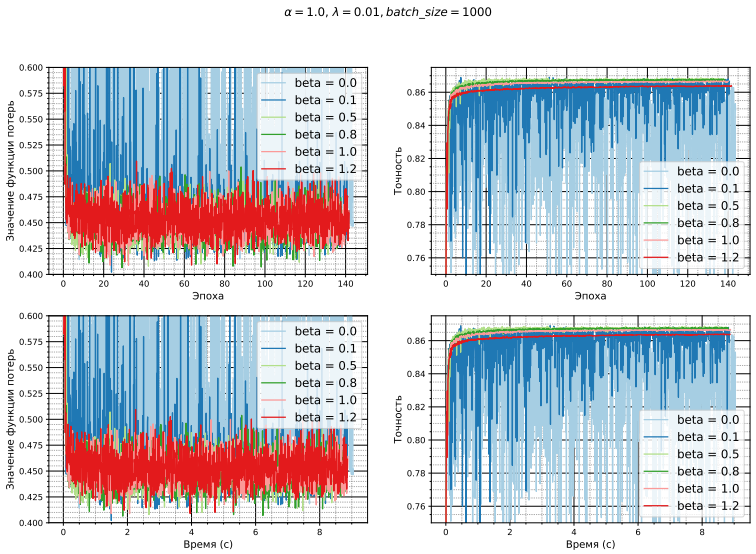


Рис. 5: Зависимость значения функции потерь и точности от времени работы и числа эпох

На графиках видно, что при маленьких значениях β скачки чрезвычайно большие, метод для таких значений не сходится. Но при больших значениях он теряет в точности. Оптимальным значением и по скачкам, и по точности, является $beta = 0.5$ (осцилляций мало, они на начальных итерациях, а точность самая большая).

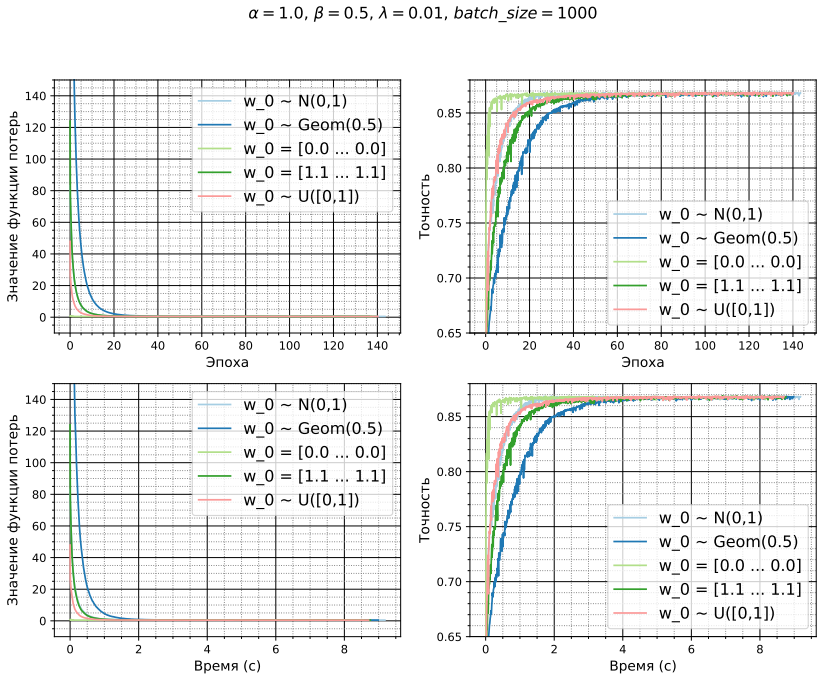


Рис. 6: Зависимость значения функции потерь и точности от времени работы и числа эпох

При возрастании числа эпох все начальные приближения показывают одинаковую точность. Но, если смотреть на начальные итерации, то самая высокая точность

достигается на нулевых весах, даже несмотря на небольшие скачки. Поэтому в качестве начального приближения выберем нулевое приближение.

Теперь посмотрим на размер батча:

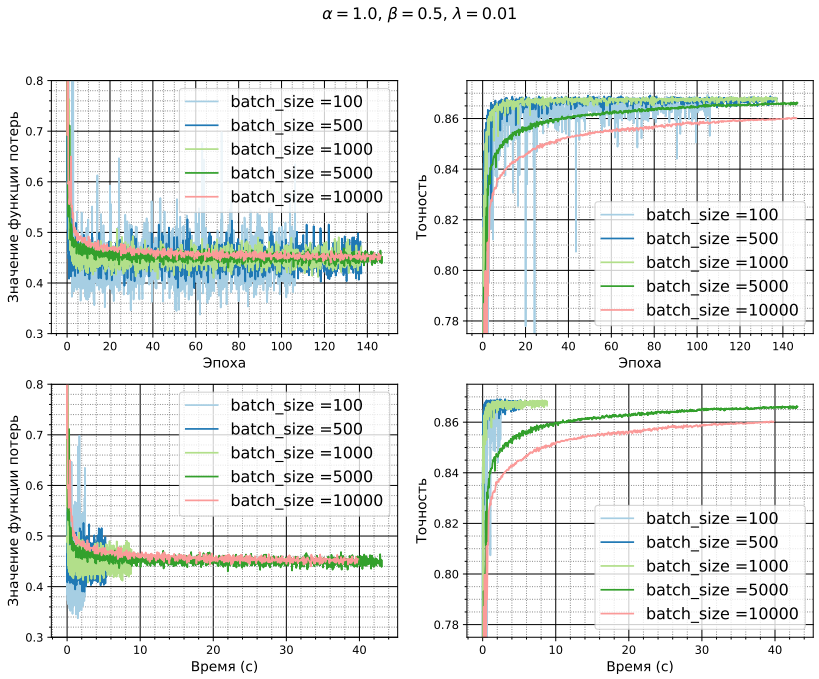
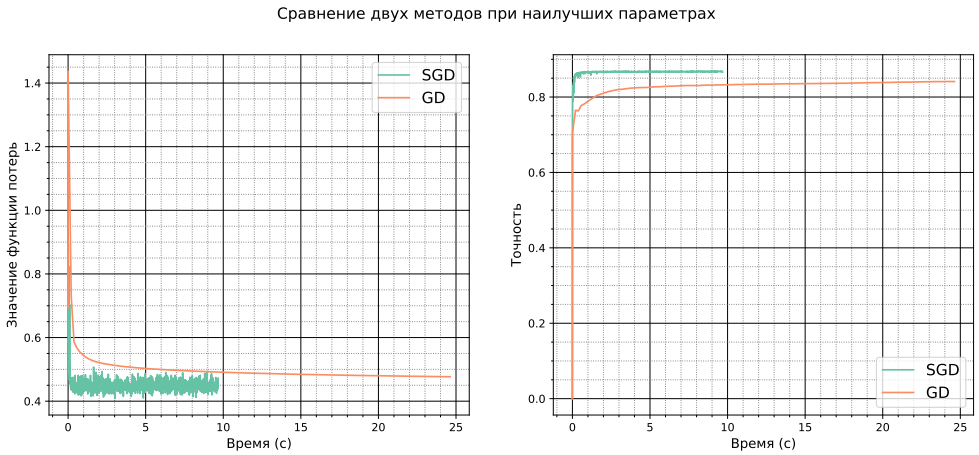


Рис. 7: Зависимость значения функции потерь и точности от времени работы и числа эпох

Отсюда видно, что при слишком больших размерах батча очень сильно увеличивается время работы метода. Также при этом уменьшается точность. С другой стороны, слишком маленьких размер батча тоже брать нельзя, потому что на графиках видны очень сильные скачки как на точности, так и на функции потерь. Оптимальным для наших данных является размер батча, равный 1000: при таком размере достигается хорошая точность за небольшое время.

3.4 Сравнение ГС и СГС

Сравним градиентный спуск и стохастический градиентный спуск при $\alpha = 1.0$, $\beta = 0.5$, $w_0 = [0.0, \ldots, 0.0]$, $\lambda = 0.01$, $batch\_size = 1000$:



Видно, что стохастический градиентный спуск показывает лучшие результаты, чем обыкновенный: он быстрее заканчивает работать, и у него выше точность.

3.5 Лемматизация

В этом эксперименте были удалены стоп-слова и применена лемматизация. Результаты отражены в таблице ниже:

Лемматизация	Число признаков	Точность на ГС	Точность на СГС
Да	14308	0.852	0.875
Нет	16050	0.841	0.866

Таблица 1: Результаты применения лемматизации

После лемматизации немного уменьшилась размерность признакового пространства, в результате чего возрасла точность. Это же иллюстрирует следующий график:

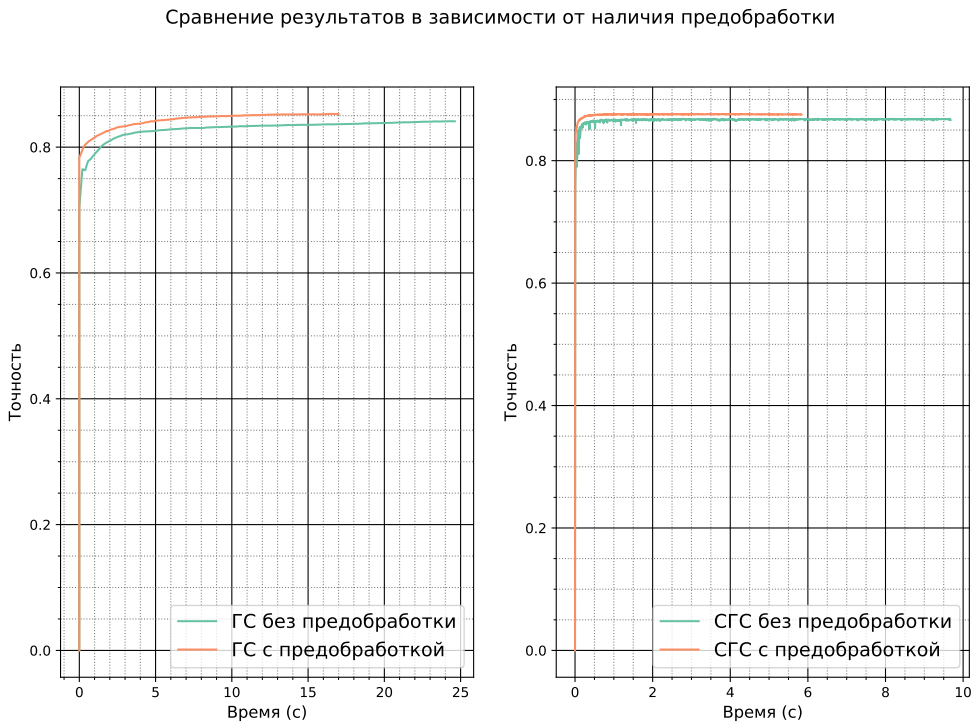


Рис. 8: Результаты применения лемматизации

Отсюда видно, что после лемматизации не только возросла точность классификации, но и уменьшилось время работы алгоритма.

3.6 Исследование BagOfWords и Tfidf

В этом эксперименте рассмотрим 2 подхода: BagOfWords и Tfidf. Варьируя параметры min_df и max_df, будем исследовать качество работы алгоритма.

min_df/max_df	0.9	0.99	0.999
0.01	456 / 456	456 / 456	456 / 456
0.001	3400/3400	3400/3400	3400/3400
0.0001	14308 /14308	14308 / 14308	14308 / 14308

Таблица 2: Размерность признакового пространства (BOW/Tdidf)

Получилось, что на размерность признакового пространства повлиял только параметр min_df: чем он меньше, тем больше у нас признаков.
Теперь посмотрим, как эти параметры влияют на точность:

min_df/max_df	0.9	0.99	0.999
0.01	0.838/ 0.800	0.838/ 0.800	0.838 / 0.800
0.001	0.873/0.811	0.873/0.811	0.873/0.811
0.0001	0.875 /0.809	0.875 / 0.809	0.875/ 0.809

Таблица 3: Точность на СГС (BOW/Tdidf)

Из этой таблицы следует, что max_df совсем не влияет на точность. Параметр min_df оказывает влияние, но зависимость установить уже сложно. Зато видно, что стратегия Tdidf пользы не приносит, точность она не повышает. Для дальнейшего использования выберем $min\_df = 0.0001$.

3.7 Лучший алгоритм

В предыдущих экспериментах мы рассмотрели все параметры, кроме параметра регуляризации. Исследуем его и построим наилучшую модель.

λ	Точность
0.0	0.895
0.01	0.875
0.05	0.853
0.1	0.841
0.5	0.802
0.8	0.800
1.0	0.784

Таблица 4: Точность при различных коэффициентах регуляризации

Из таблицы следует, что лучше взять $\lambda = 0.01$.
При выбранных нами параметрах стохастический метод градиентного спуска показывает на тестовой выборке точность 0.855. Ошибки проанализируем ниже.

3.8 Добавление n-грамм

Используя конструктор TfidfVectorizer, добавим n-граммы. Результаты получились следующие:

n-грамма	Точность	Время работы
(1 1)	time = 8.41	acc = 0.763
(1 2)	time = 13.62	acc = 0.750
(1 3)	time = 16.09	acc = 0.748
(1 4)	time = 17.02	acc = 0.748
(2 2)	time = 3.80	acc = 0.775
(2 3)	time = 3.14	acc = 0.777
(2 4)	time = 4.00	acc = 0.777
(3 3)	time = 1.27	acc = 0.757
(3 4)	time = 1.62	acc = 0.755
(4 4)	time = 2.04	, acc = 0.661

Таблица 5: Результат добавления n-грамм

Из таблицы следует, что добавление n-грамм не улучшает точность алгоритма, поэтому для наших данных использовать их не будем.

4 Вывод

В результате проведение экспериментов было выяснено, что для наших данных лучше всего подходит стохастический градиентный спуск с параметрами $\alpha = 1.0$, $\beta = 0.5$, $\lambda = 0.01$, $batch\_size = 1000$. Также нужно применить лемматизацию. В результате на тестовой выборке алгоритм показал точность 0.8558. Проанализируем некоторые ошибки:

1. aapn bhtla aanand jhala — для этого предложения нет перевода, слова с опечатками. Модель отнесла этот комментарий к токсичным.

2. hell justin — модель отнесла этот комментарий к токсичным, хотя ясно, что здесь просто опечатка в первом слове.

3. hello everyone tell freak — модель классифицировала этот комментарий как нетоксичный, вероятно, из-за того, что у последнего слова есть несколько значений, в том числе вполне приемлемые.

4. dear god site horrible — на мой взгляд, этот комментарий можно назвать как токсичным, так и нет. В тонкой ситуации у модели не получилось классифицировать правильно

Таким образом, наша модель в основном ошибается в комментариях с опечатками или в комментариях, которые могут быть двояко классифицированы.