«Московский государственный университет имени М. В. Ломоносова»
Факультет вычислительной математики и кибернетики
Кафедра методов математического прогнозирования

Отчет по практическому заданию
Задание 2. Градиентные методы обучения линейных моделей

Дмитриева Татьяна, 3 курс

Октябрь 2021

Содержание

1 Введение

В данном задании предлагается написать собственную реализацию линейного классификатора на языке Python и провести серию экспериментов с его использованием. Целью данных исследований является нахождение оптимальных параметров для используемого классификатора, при которых достигается более высокий процент правильно предсказанных ответов на тестовых данных. Эксперименты этого задания проводятся на датасете, состоящим из комментариев раздела обсуждений английской Википедии, которые разделили на две категории в соответствии с тем, являются ли они токсичными.

2 Теоретические выкладки

2.1 Бинарная логистическая регрессия

Функция потерь для бинарной логистической регрессии:

$$Q(X, w) = \frac{1}{\ell}\sum_{i=1}^{\ell}\big(\log\left(1 + \exp(-y_i\langle w, x_i\rangle\right))$$

Градиент функции потерь для задачи бинарной логистической регрессии:

$$dQ_w = \frac{1}{\ell}\sum_{i=1}^{\ell}\frac{-d(y_i\langle w, x_i\rangle)\exp(-y_i\langle w, x_i\rangle)}{\big(\log\left(1 + \exp(-y_i\langle w, x_i\rangle\right)\big)} = \frac{1}{\ell}\sum_{i=1}^{\ell}\frac{-(y_i\langle dw, x_i\rangle)\exp(-y_i\langle w, x_i\rangle)}{\big(\log\left(1 + \exp(-y_i\langle w, x_i\rangle\right)\big)}$$

Тогда

$$\nabla Q_w(X, w) = \frac{1}{\ell}\sum_{i=1}^{\ell}\frac{y_i x_i \exp(-y_i\langle w, x_i\rangle)}{\big(\log\left(1 + \exp(-y_i\langle w, x_i\rangle\right)\big)}$$

2.2 Мультиномиальная логистическая регрессия

Функция потерь для мультиномиальной логистической регрессии:

$$Q(X, w) = -\frac{1}{\ell}\sum_{i=1}^{\ell}\log\left(\frac{exp(\langle w_{y_i}, x_i\rangle)}{\sum_{t=1}^{K} exp(\langle w_t, x_i\rangle)}\right)$$

Градиент функции потерь для задачи мультиномиальной логистической регрессии:

$$\frac{\partial Q}{\partial w_k} = -\frac{1}{\ell}\sum_{i=1}^{\ell}\frac{\sum_{t=1}^{K} exp(\langle w_t, x_i\rangle)}{exp(\langle w_{y_i}, x_i\rangle)}\left(\frac{\frac{\partial}{\partial w_k}\big(exp(\langle w_{y_i}, x_i\rangle)\big)\sum_{t=1}^{K} exp(\langle w_t, x_i\rangle) - exp(\langle w_{y_i}, x_i\rangle)\frac{\partial}{\partial w_k}\big(\sum_{t=1}^{K} exp(\langle w_t, x_i\rangle)\big)}{\big(\sum_{t=1}^{K} exp(\langle w_t, x_i\rangle)\big)^2}\right) =$$

$$= -\frac{1}{\ell}\sum_{i=1}^{\ell} I[y_i = k]\left(\frac{\big(exp(\langle w_k, x_i\rangle)\frac{\partial}{\partial w_k}(\langle w_k, x_i\rangle)\big)\sum_{t=1}^{K} exp(\langle w_t, x_i\rangle) - exp(\langle w_k, x_i\rangle)exp(\langle w_k, x_i\rangle)\frac{\partial}{\partial w_k}(\langle w_k, x_i\rangle)\big)}{\big(\sum_{t=1}^{K} exp(\langle w_t, x_i\rangle)\big)exp(\langle w_k, x_i\rangle)}\right) -$$

$$-\frac{1}{\ell}\sum_{i=1}^{\ell} I[y_i \neq k]\left(\frac{-exp(\langle w_{y_i}, x_i\rangle)exp(\langle w_k, x_i\rangle)\frac{\partial}{\partial w_k}(\langle w_k, x_i\rangle)\big)}{\big(\sum_{t=1}^{K} exp(\langle w_t, x_i\rangle)\big)exp(\langle w_{y_i}, x_i\rangle)}\right) =$$

$$= \frac{1}{\ell}\sum_{i=1}^{\ell} I[y_i = k]\left(\frac{\big(-\frac{\partial}{\partial w_k}(\langle w_k, x_i\rangle)\big)\sum_{t=1}^{K} exp(\langle w_t, x_i\rangle) + exp(\langle w_k, x_i\rangle)\frac{\partial}{\partial w_k}(\langle w_k, x_i\rangle)\big)}{\big(\sum_{t=1}^{K} exp(\langle w_t, x_i\rangle)\big)}\right) +$$

$$+\frac{1}{\ell}\sum_{i=1}^{\ell} I[y_i \neq k]\left(\frac{exp(\langle w_k, x_i\rangle)\frac{\partial}{\partial w_k}(\langle w_k, x_i\rangle)\big)}{\big(\sum_{t=1}^{K} exp(\langle w_t, x_i\rangle)\big)}\right) =$$

$$= \frac{1}{\ell}\sum_{i=1}^{\ell} -I[y_i = k]\frac{\partial}{\partial w_k}(\langle w_k, x_i\rangle) + \frac{1}{\ell}\sum_{i=1}^{\ell}\left(\frac{exp(\langle w_k, x_i\rangle)\frac{\partial}{\partial w_k}(\langle w_k, x_i\rangle)\big)}{\big(\sum_{t=1}^{K} exp(\langle w_t, x_i\rangle)\big)}\right) =$$

Тогда

$$\nabla Q_w(X, w) = -\frac{1}{\ell}\sum_{i=1}^{\ell}\left(I[y_i = k]x_i + \frac{exp(\langle w_k, x_i\rangle)x_i}{\big(\sum_{t=1}^{K} exp(\langle w_t, x_i\rangle)\big)}\right)$$

2.3 Мультиномиальная логистическая регрессия с количеством классов - 2

Приведение задачи мультиномиальной логистической регрессии с K = 2 к бинарной логистической регрессии. Запишем функцию потерь для бинарной логистической регрессии и для мультиномиальной с K = {-1, 1}.

$$Q(X,w)_b = \frac{1}{\ell}\sum_{i=1}^{\ell}\big(\log\left(1+\exp(-y_i\langle w,x_i\rangle\right)) = -\frac{1}{\ell}\sum_{i=1}^{\ell}\big(\log\frac{\exp(y_i\langle w,x_i\rangle)}{(1+\exp(y_i\langle w,x_i\rangle))}$$

$$Q(X,w)_2 = -\frac{1}{\ell}\sum_{i=1}^{\ell}\log\left(\frac{exp(\langle w_{y_i},x_i\rangle)}{exp(\langle w_{-1},x_i\rangle)+exp(\langle w_1,x_i\rangle)}\right) = -\frac{1}{\ell}\sum_{i=1}^{\ell}\log\frac{I[y_i=1]exp(\langle w_1,x_i\rangle)+I[y_i=-1]exp(\langle w_{-1},x_i\rangle)}{exp(\langle w_{-1},x_i\rangle)+exp(\langle w_1,x_i\rangle)}$$

Сделаем замену для мультиномиальной логистической регрессии $w_1 - w_{-1} = w$.

$$Q(X,w)_2 = -\frac{1}{\ell}\sum_{i=1}^{\ell}\log\left(\frac{I[y_i=1]exp(\langle w,x_i\rangle)}{1+exp(\langle w,x_i\rangle)}\right) - \frac{1}{\ell}\sum_{i=1}^{\ell}\log\frac{I[y_i=-1]exp(\langle -w,x_i\rangle)}{1+exp(\langle -w,x_i\rangle)} = -\frac{1}{\ell}\sum_{i=1}^{\ell}\big(\log\frac{\exp(y_i\langle w,x_i\rangle)}{(1+\exp(y_i\langle w,x_i\rangle))}$$

Отсюда можно сделать вывод, что при соответствующей замене задача мультиномиальной логистической регрессии сводится к биномиальной логистической регрессии.

3 Эксперименты

3.1 1 эксперимент

3.1.1 Описание эксперимента

В данном эксперименте требовалось произвести предварительную обработку текста, приведя его в нижнему регистру и заменив в нем все символы, не являющиеся буквами и цифрами, на пробелы. Для этого были использованы следующие функции языка Python: **str.lower** для приведения к нижнему регистру и **re.sub** для требуемой замены не интересующих нас символов на пробелы.

3.1.2 Вывод

В ходе такой обработки информативность текста, вероятно, уменьшилась, ведь мы убрали знаки пунктуации и другие, важные для оценки токсичности комментариев, символы. Тем не менее хранить такую информацию не очень полезно, ведь в ней также будут содержаться избыточные или даже ошибочные данные, которые лишь усложнят анализ текста.

3.2 2 эксперимент

3.2.1 Описание эксперимента

В ходе второго эксперимента необходимо было преобразовать наши данные в удобный для анализирования формат - разреженную матрицу, которая будет хранить информацию о том, какие слова и в каком количестве встретились в каждом из наших документов. Для данного преобразования мы воспользовались методами **fit_transform** и **fit** класса **CountVectorizer** языка Python. При конструировании этого класса параметр **min_df** был установлен равным 0.001. Он отвечал за минимальную частоту использования слова в документах из тренировочной выборки, при которой это слово вносилось в признаковое пространство нашей разреженной матрицы. Мы использовали этот параметр для того, чтобы уменьшить вероятность переобучения классификатора на редко встречающихся словах, а также увеличить скорость обучения алгоритма, уменьшив признаковое пространство тренировочной выборки.

3.2.2 Вывод

После преобразования нашего текста мы смогли получить информацию в форме векторов, что позволило использовать ее для обучения реализованного нами классификатора. Разумеется, такие преобразования привели к потере данных о том, в каком контексте использовалось то или иное слово. Это довольно важная информация, которая могла улучшить распознавание токсичных комментариев. Однако она также могла сильно усложнить задачу минимизации функции потерь нашего алгоритма и уменьшить эффективность его работы, что не оправдывает ее использование в нашем случае.

3.3 3 эксперимент

3.3.1 Описание эксперимента

В третьем эксперименте требовалось изучить поведение градиентного спуска для задачи логистической регрессии в зависимости от используемых для этого параметров:

- размера шага step_alpha
- размера шага step_beta
- начального приближения

Для изучения поведения градиентного спуска при различных step_alpha мы построили зависимости значения функции потерь и **accuracy** от количества итераций и времени работы алгоритма, использовав для его конструирования следующие значения параметров **step_beta = 0.1**, **tolerance = 1e-7**, **max_iter = 1000**, **l2_coef = 0.0001**, **w0 = 0**.

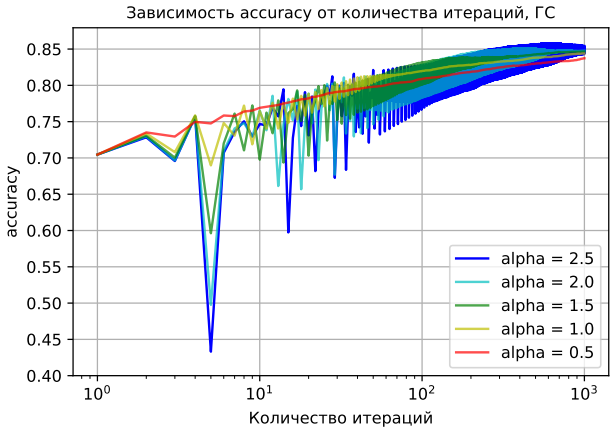

Рис. 1: Градиентный спуск

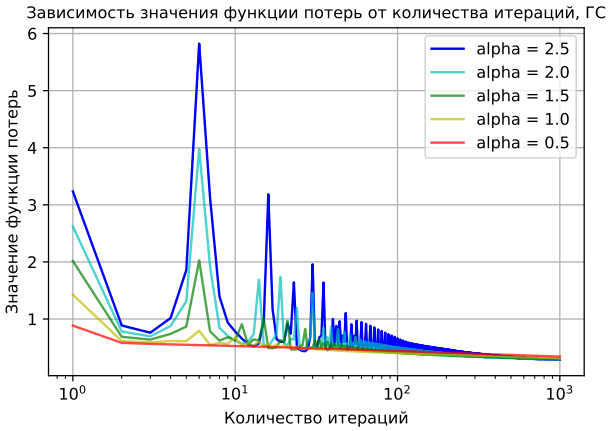

Рис. 2: Градиентный спуск

Для изучения поведения градиентного спуска при различных step_beta были построены зависимости значения функции потерь и **accuracy** от количества итераций и времени работы алгоритма со следующими значения параметров **step_alpha = 0.5**, **tolerance = 1e-7**, **max_iter = 1000**, **l2_coef = 0.0001**, **w0 = 0**.

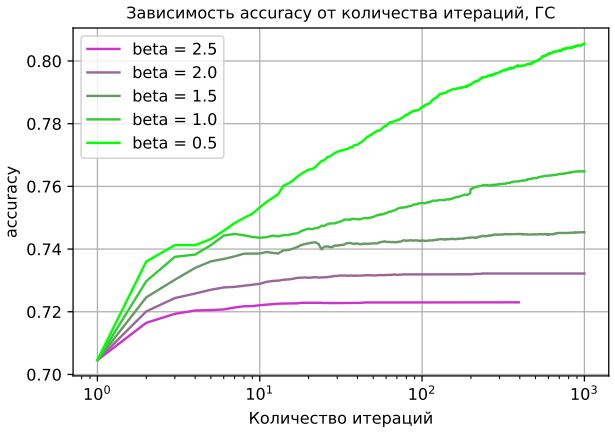

Рис. 3: Градиентный спуск

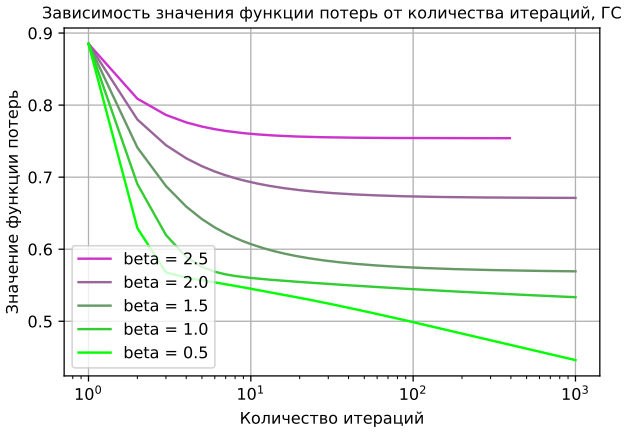

Рис. 4: Градиентный спуск

Для изучения поведения градиентного спуска при различных начальных условиях были построены зависимости значения функции потерь и **accuracy** от количества итераций и времени работы алгоритма со следующими значения параметров **step_alpha = 0.5**, **step_beta = 0.1**, **tolerance = 1e-7**, **max_iter = 1000**, **l2_coef**

= **0.0001**. Для начальных приближений использовались нулевые веса; единичные веса; веса, взятые из нормального распределения со средним - 0 и стандартным отклонением - 0.3; веса, взятые случайно из множества {-1/2N, 1/2N}; веса, равные 1/N. Здесь N - размерность признакового пространства.

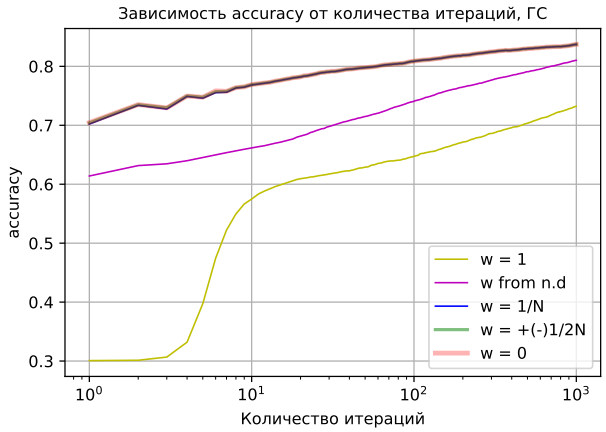

Рис. 5: Градиентный спуск

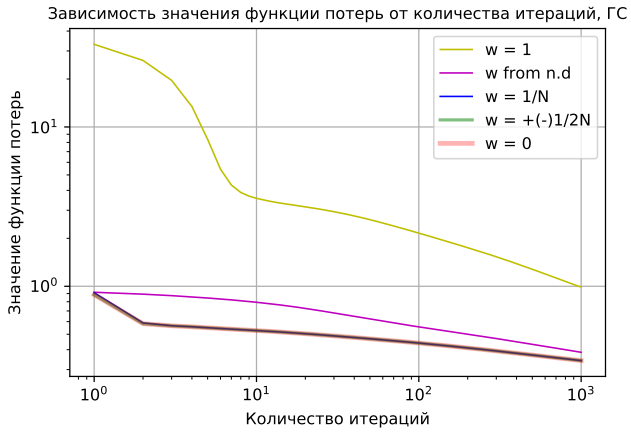

Рис. 6: Градиентный спуск

Во всех построенных в данном эксперименте графиках мы не учитывали начальную точку, так как ее наличие ухудшало интерпретируемость графиков, потому что **accuracy** было слишком низким. Этого же решения будем придерживаться в дальнейших экспериментах.

3.3.2 Вывод

На графике зависимости **accuracy** от количества итераций для градиентного спуска(рис.1) для различных step_alpha мы видим, что итоговые значения точности для каждого из step_alpha близки друг к другу. Однако при увеличении величины step_alpha **accuracy** на малых итерациях сильно изменяется. Это связано с тем, что при увеличении step_alpha темп обучения тоже увеличивается, именно поэтому на графике мы видим такие резкие скачки. Зеркальная ситуация наблюдается с графиком зависимости значения функции потерь от количества итераций(рис.2). Это объясняется тем, что между функцией потерь и точностью распознавания классификатора есть связь, которую мы использовали при реализации, - при уменьшении значения функции потери точность должна увеличиваться. Оба графика подтверждают эту связь.

На графике зависимости **accuracy** от количества итераций для градиентного спуска(рис.3) при различных step_beta можно заметить, что значение точности значительно отличается для каждого из них. Наблюдается самое высокое качество распознавания у классификатора при step_beta = 0.5 и самое низкое при step_beta = 2.5. Это подтверждает график зависимости значения функции потерь от количества итераций(рис.4). Кроме того, из этих графиков можно сделать вывод, что при уменьшении step_beta точность метода вырастает, а значение функции потерь уменьшается. Это следует из того, что при увеличении step_beta темп обучения падает быстрее, что приводит значение функции потерь к локальному минимуму, не давая возможности отыскать более оптимальное значение.

На графике зависимости **accuracy** и функции потерь от количества итераций для градиентного спуска(рис.5, рис.6) при различных начальных приближениях можно заметить, что наилучший результат работы метода достигается при выборе нулевого начального приближения или близкого к нулевому, но **accuracy** становится хуже при удалении начальных условий от нулевых. Можно предположить, что это происходит вследствие того, что признаковое пространство наших объектов достаточно велико и для каждого из объектов вектор признаков в основном состоит из нулей, поэтому для определения токсичности комментария мы используем веса с очень малыми значениями, близкими к нулевым.

Графики зависимости **accuracy** и функции потерь от времени работы методов при различных step_alpha, step_beta, начальных приближений приведены в приложении(рис.37-42), так как они очень похожи на приведенные здесь графики. Таким образом, мы можем сделать вывод, что каждая из итераций метода, вероятно, выполняется за одинаковое время.

3.4 4 эксперимент

3.4.1 Описание эксперимента

В четвертом эксперименте необходимо было изучить поведение стохастического градиентного спуска для задачи логистической регрессии в зависимости от используемых для этого параметров:

- размера шага step_alpha
- размера шага step_beta
- начального приближения
- размера подвыборки batch_size

Для всех приведенных ниже графиков будем использовать следующие параметры(за исключением тех, относительно которых ищем зависимости): **step_alpha = 0.5**, **step_beta = 0.1**, **tolerance = 1e-7**, **max_iter = 1000**, **l2_coef = 0.0001**, **w0 = 0**, **log_freq = 0.2**.

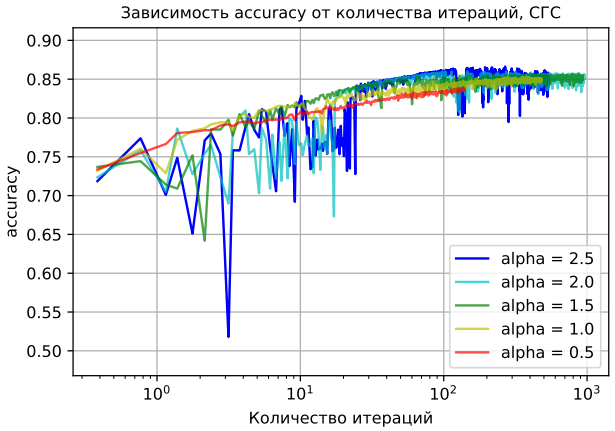

Рис. 7: Стохастический градиентный спуск

Рис. 8: Стохастический градиентный спуск

На приведенных выше графиках(рис.7, рис.8) можно заметить, что при изменении step_alpha итоговое значение, как и для обычного градиентного спуска, почти не меняется. Однако колебания **accuracy** и функции потерь более сильны, ведь мы за одну эпоху делаем несколько сдвигов в отличие от ГС.

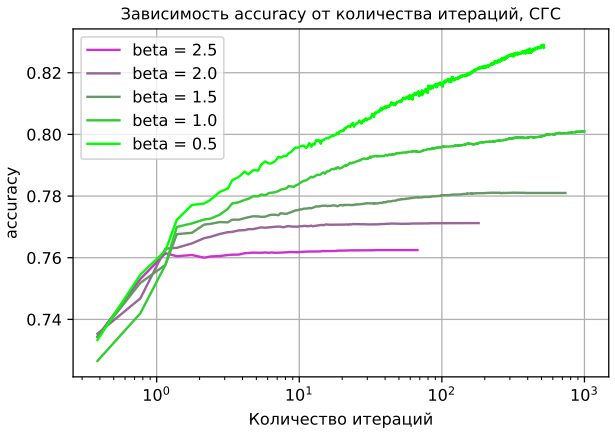

Рис. 9: Стохастический градиентный спуск

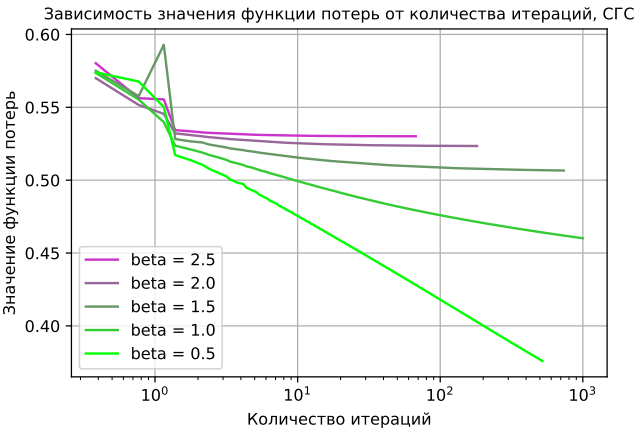

Рис. 10: Стохастический градиентный спуск

При step_beta для стохастического градиентного спуска мы можем заметить аналогичную связь уменьшения точности алгоритма с увеличением step_beta, как и для обычного градиентного спуска, однако графики(рис.9, рис.10) не такие гладкие, что также объясняется большим количеством шагов и изменением исследуемых объектов на каждой итерации.

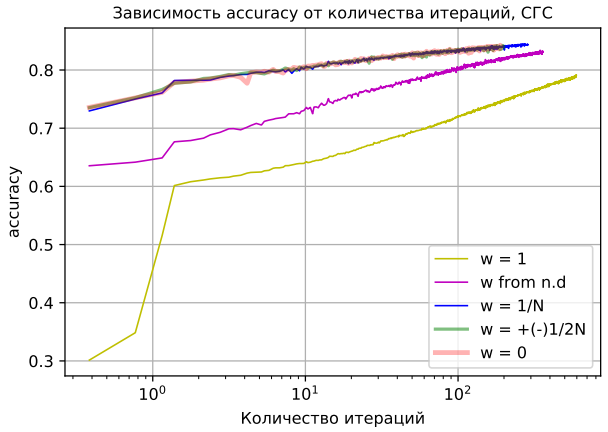

Рис. 11: Стохастический градиентный спуск

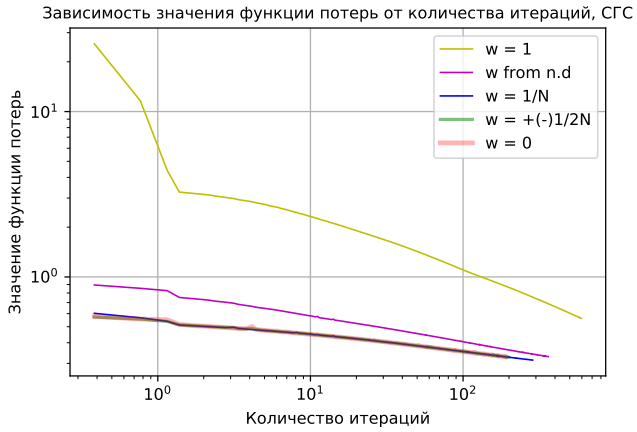

Рис. 12: Стохастический градиентный спуск

Для исследования **accuracy** использовались те же значения для начального приближения, как и для обычного градиентного спуска в эксперименте 3. На графиках(рис.11, рис.12) можно заметить, что с отдалением от нулевого начального приближения качество ухудшается. Однако для стохастического градиентного спуска чуть более хорошее качество достигается при начальных условиях - 1/N.

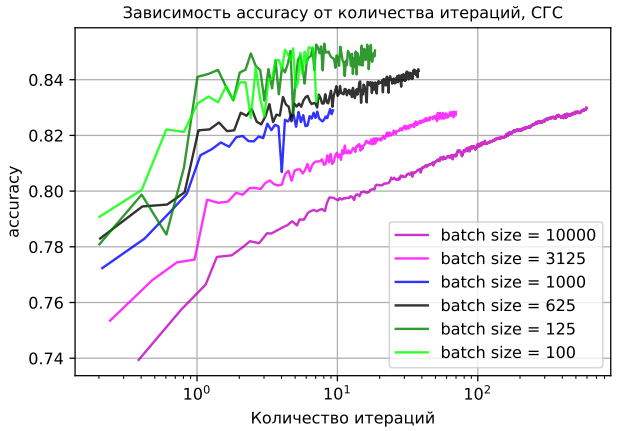

Рис. 13: Стохастический градиентный спуск

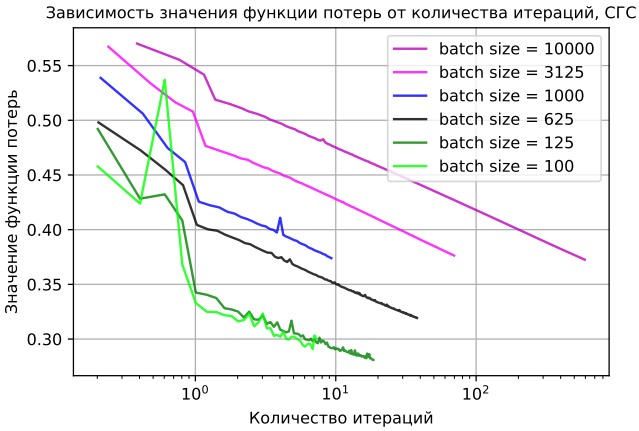

Рис. 14: Стохастический градиентный спуск

Рассматривая графики зависимости значения функции потерь и **accuracy** от размера подвыборки batch_size(рис.13-16), мы можем увидеть что для размеров батчей до 1000 наблюдается увеличение точности с увеличением времени работы и уменьшением размера батчей. Однако для довольно больших размеров батчей точность понижается по сравнению с маленькими размерами батчей, но время работы при этом начинает возрастать.

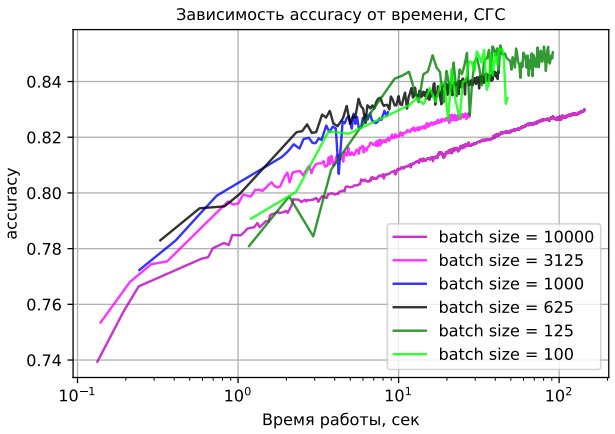

Рис. 15: Стохастический градиентный спуск

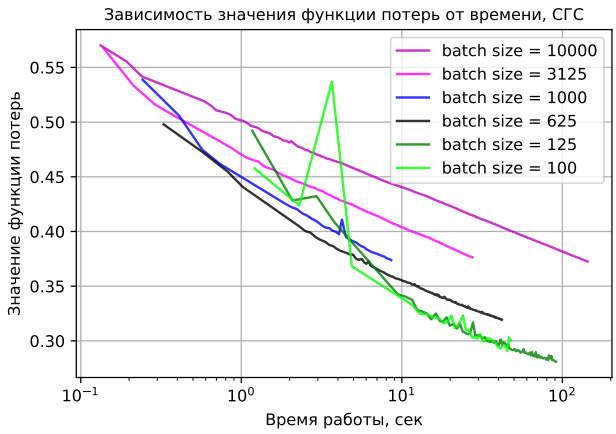

Рис. 16: Стохастический градиентный спуск

3.4.2 Вывод

Таким образом, то, что мы видим на графиках, в общих чертах похоже на то, что мы наблюдали в эксперименте 3, однако каждый из графиков колеблется чаще, чем такой же график у обычного градиентного спуска. Как было сказано выше, это следует из того, что шагов, совершаемых алгоритмом, происходит больше.

Для размеров батчей наблюдаемые зависимости, вероятно, указывают на то, что маленькие батчи позволяют выявить более тонкие зависимости и повысить качество, однако слишком большие батчи приближают наш алгоритм к обычному градиентному спуску, который, в среднем, показывает чуть менее хороший результат, чем стохастический градиентный спуск.

Графики для **accuracy** и функции потерь от времени работы методов при различных step_alpha, step_beta, начальных приближений приведены в приложении(рис.43-48), так как они очень похожи на приведенные здесь графики. Также как и в 3 эксперименте, мы можем предположить, что каждая из итераций метода, вероятно, выполнялась за одинаковое время.

3.5 5 эксперимент

3.5.1 Описание эксперимента

В этом эксперименте мы должны сравнить стохастический и обычный градиентный спуск. Для этого мы сначала выберем наилучшие параметры для обоих методов, а затем сравним их. Для поиска наилучших параметров была построена матрица зависимости **accuracy** от параметров step_beta и step_alpha.(рис.17, рис.18). Здесь исследуемые параметры были взяты из предположения, что наилучшие step_beta достаточно малы, а наиболее эффективные step_alpha при приблизительно равных значениях точности и более быстрых подсчетах лежат около единицы.

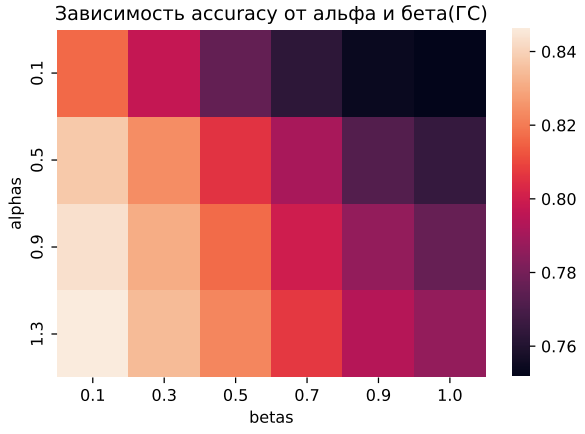

Рис. 17: Точность

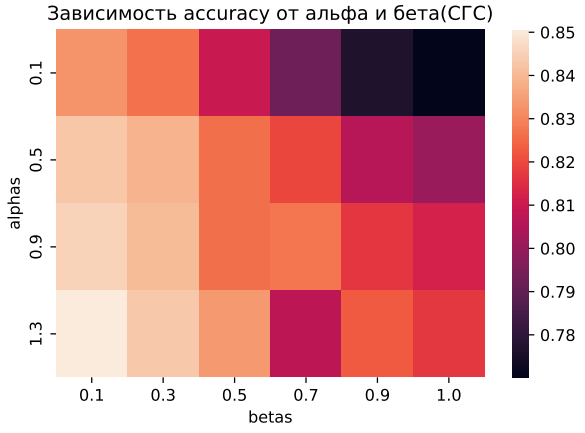

Рис. 18: Точность

Как для стохастического, так и для обычного градиентного спуска наилучшим сочетанием параметров оказались step_alpha = 1.3, step_beta = 0.1. Сравним два этих метода по совершаемым ими ошибкам, времени работы, значению функции ошибок и **accuracy** на последней итерации работы.

Таблица ошибок(ГС)
(Ответ/Предсказание)

	-1	0	1
-1	12099	267	2067
1	827	24	5392

Рис. 19: Ошибки

Таблица ошибок(СГС)
(Ответ/Предсказание)

	-1	0	1
-1	12129	267	2037
1	790	24	5429

Рис. 20: Ошибки

Таблица
сравнения

Метод	Время, сек	accuracy	ФП
SGD	42.192	0.84919	0.29
GD	84.222	0.84595	0.30

Рис. 21: Точность

3.5.2 Вывод

Таким образом, мы можем сделать вывод, что стохастический градиентный спуск работает при прочих равных условиях быстрее и его точность оказывается незначительно, но лучше(рис.21). По таблице с ошибками(рис.19, 20) мы видим, что СГС в целом более хорошо справляется с задачей классификации комментариев, правильно предсказывая большее их количество, независимо от того, являются ли они токсичными. Вероятно, это связано с тем, что он более гибкий по сравнению с обычным градиентным спуском, так как за раз анализирует лишь часть данных, при этом совершая больше итераций.

3.6 6 эксперимент

3.6.1 Описание эксперимента

В этом эксперименте от нас требуется проанализировать, как дополнительная предобработка нашего текста влияет на точность работы нашего классификатора. Для этого мы воспользуемся алгоритмом лемматизации и удалим из текста стоп-слова, используя для этого список стоп-слов из библиотеки nltk языка Python. После лемматизации размер признакового пространства составил 3043, в то время как до лемматизации он был равен 3736. Для исследования времени и точности изучим работу наших алгоритмов при **step_alpha** = 0.9, 1.3 и **step_beta** = 0.1, 0.3, 0.5, **tolerance = 1e-7**, **max_iter = 1000**, **l2_coef = 0.0001**, **w0 = 0**, **log_freq = 0.2**, **batch_size = 10000**.

Таблица времени работы
без предобработки(ГС)

	0.1	0.3	0.5
0.9	99.25	97.02	63.33
1.3	98.25	95.29	94.53

Рис. 22: Время

Таблица времени работы с
предобработкой(ГС)

	0.1	0.3	0.5
0.9	61.51	62.17	63.33
1.3	62.93	59.63	64.97

Рис. 23: Время

Таблица точности без
предобработки(СГС)

	0.1	0.3	0.5
0.9	0.847	0.837	0.835
1.3	0.851	0.843	0.833

Рис. 24: Точность

Таблица точности с
предобработкой(СГС)

	0.1	0.3	0.5
0.9	0.850	0.849	0.844
1.3	0.851	0.849	0.846

Рис. 25: Точность

Другие графики(рис.49-52) приведены в приложении. Это связано с тем, что точность у стохастического градиентного спуска в среднем выше, поэтому оказывается более интересным сравнить точность именно на СГС классификаторе. В то же время длительность работы стоит сравнить у обычных градиентных спусков, потому что при стохастическом ГС не всегда по времени можно однозначно сделать какие-то выводы.

3.6.2 Вывод

Исходя из полученных нами результатов, можно сказать, что в случае предобработки градиентный спуск начинает работать в среднем быстрее. Это следует из того, что мы уменьшили наше признаковое простран-

ство, тем самым уменьшив количество вычислений в нашем классификаторе.(рис.22-23). Точность при этом выросла(рис.24-25). Это объясняется тем, что мы с помощью предобработки убрали избыточную информацию и повысили информативность каждого из признаков.

3.7 7 эксперимент

3.7.1 Описание эксперимента

В данном эксперименте требовалось изучить поведение алгоритма в зависимости от следующих параметров:

- использовалось представление **BagOfWords** или **Tfidf**
- параметров **min_df** и **max_df** конструкторов

Для изучения зависимости работы алгоритма от используемого представления данных обучим наши классификаторы при **step_alpha** = 0.9, 1.3 и **step_beta** = 0.1, 0.3, 0.5, **tolerance** = **1e-7**, **max_iter** = **1000**, **l2_coef** = **0.0001**, **w0** = **0**, **log_freq** = **0.33**, **batch_size** = **1000**.

Таблица времени работы(ГС, BagOfWords)

	0.1	0.3	0.5
0.9	99.25	97.02	63.33
1.3	98.25	95.29	94.53

Рис. 26: Время

Таблица времени работы(ГС, Tfidf)

	0.1	0.3	0.5
0.9	85.12	90.09	83.97
1.3	83.97	81.99	84.34

Рис. 27: Время

Таблица точности(СГС, BagOfWords)

	0.1	0.3	0.5
0.9	0.847	0.837	0.835
1.3	0.851	0.843	0.833

Рис. 28: Точность

Таблица точности(СГС, Tfidf)

	0.1	0.3	0.5
0.9	0.849	0.844	0.835
1.3	0.844	0.843	0.841

Рис. 29: Точность

Когда мы использовали представление данных **Tfidf**, размер признакового пространства не изменился при том же значении min_df = 0.001, что и для первоначального представления **BagOfWords**. Он составил 3736. Это ожидаемый результат, ведь данные, с которыми мы работаем, не изменились. Однако вместо количества слов мы стали использовать их частоту.

Сравнивая точность при использовании представления **Tfidf** и при **BagOfWords**(рис.28-29) можно сделать вывод о том, что в некоторых случаях точность немного возросла, а в некоторых упала. Из этого можно сделать вывод, что эти представления работают хорошо при правильно подобранных параметрах, которые отличаются для каждого из них. То же самое можно сказать о времени работы(рис.26-27). Однако на тех параметрах, которые мы использовали, более быстрым оказалось представление **Tfidf**.

Здесь, как и в 6 эксперименте, мы оставили таблицу для сравнения точности у стохастического градиентного спуска(рис.28, 29) и таблицу для сравнения времени работы обычного градиентного спуска(рис.27, 27) по тем же соображениям. Другие таблицы отнесены в приложение(рис.53-56).

Для изучения работы алгоритмов при изменении параметров min_df max_df были построены классификаторы со следующим заданием аргументов для конструктора: **step_alpha** = **1.3**, **step_beta** = **0.1**, **tolerance** = **1e-7**, **max_iter** = **1000**, **l2_coef** = **0.0001**. Веса в начальном приближении были нулевыми.

Таблица точности

min_df	1e-5	1e-4	1e-3	1e-2	1e-1
GD	0.853	0.851	0.846	0.810	0.713
SGD	0.860	0.849	0.854	0.810	0.710

Рис. 30: Точность

Таблица точности

max_df	1.00	0.20	0.15	0.10	0.05	0.01
GD	0.853	0.854	0.856	0.854	0.855	0.795
SGD	0.860	0.857	0.855	0.857	0.858	0.792

Рис. 31: Точность

Сравнивая точность работы наших алгоритмов при различных значениях параметров(рис.30-31), отвечающих за частоту использования слов в документах, было установлено, что при увеличении min_df точность

уменьшается, а при увеличении max_df увеличивается. В обоих случаях это соответствует увеличению признакового пространства, то есть усложняет наш алгоритм, при этом повышая его качество.

Таблица времени

min_df	1e-5	1e-4	1e-3	1e-2	1e-1
GD	126.90	109.47	83.90	64.17	34.44
SGD	96.90	73.43	57.28	236.71	418.95

Рис. 32: Время

Признаковое пространство max_df

1.00	0.20	0.15	0.10	0.05	0.01
89368	89348	89331	89313	89260	88800

Рис. 33: Признаковое пространство

Оценив время работы нашего алгоритма при различных max_df и min_df(рис.32, 34) можно получить предсказуемый результат - при уменьшении количества признаков время уменьшается, потому что алгоритм становится проще, а количество вычислений меньше.

Таблица времени

max_df	1.00	0.20	0.15	0.10	0.05	0.01
GD	128.38	123.80	123.77	123.54	124.08	124.08
SGD	303.13	171.02	190.06	128.55	271.40	471.39

Рис. 34: Время

Признаковое пространство min_df

1e-5	1e-4	1e-3	1e-2	1e-1
89368	16050	3736	568	55

Рис. 35: Признаковое пространство

3.7.2 Вывод

Таким образом от представления данных зависит то, в каких условиях будет лучше работать наш алгоритм и с какими параметрами будет получен наилучший результат. Признаковое пространство же напрямую влияет на точность классификации объектов при работе методов, так как их сложность возрастает, а хранимая и используемая для работы информация увеличивается.

3.8 8 эксперимент

3.8.1 Описание эксперимента

Здесь предлагается выбрать наилучший из исследованных нами алгоритмов и подобрать лучшие параметры. Для этого мы решили остановиться на стохастическом градиентном спуске при использовании лемматизации и **step_alpha = 1.3**, **step_beta = 0.1**, **tolerance = 1e-7**, **max_iter = 1000**, **l2_coef = 0.0001**, **w0 = 1/N**, **log_freq = 0.33**, **batch_size = 125**.

В таблице ошибок(рис.36) мы видим, что почти в 4 раза больше алгоритм ошибался на токсичных комментариях, определяя их не токсичными. Однако если посмотреть на общее количество токсичных и не токсичных комментариев, можно заметить, что токсичных комментариев в нашей тестовой выборке больше, так что в процентном соотношении получилось, что алгоритм ошибся в 15% в случае с токсичными комментариями и в 8% в случае с не токсичными комментариями.

Таблица ошибок
(Ответ/Предсказание)

	-1	0	1
-1	12072	170	2191
1	502	8	5733

Рис. 36: Ошибки

Объекты, на которых были допущены ошибки

Текст	Ответ	Предсказание
<Korean text>	not toxic	nothing
father heaven could not stand looking at his six ugly destructive sons so he throw all six under mother earth in the black hole	not toxic	toxic
******How many times are you going to reply to yourself, sockpuppet?	toxic	not toxic
U You make me so horny -Archer	not toxic	toxic

При анализировании комментариев, на которых наш алгоритм совершал ошибки, было установлено, что он не может определить, токсичен или нет комментарий, если язык, на котором он был написан, - иностранный. Это объяснимо способом построения нашего представления, так как мы использовали CountVectorizer, который, вероятно, плохо поддерживает некоторые из кодировок. Также стоит отметить, что большая часть комментариев

была на английском, так что наш алгоритм не мог научиться распознавать токсичность для других языков, потому что не имел информации для тех слов, которые использовались в комментарии.

Во втором примере ошибочно распознанного комментария можно предположить, что встретились слова, которые очень часто встречались в токсичных комментариях, из-за чего их вес в сторону токсичности был очень высок. Здесь мы как раз встречаемся с озвученной ранее проблемой отсутствия хранимой информации о контексте употребления того или иного слова.

В третьем примере наш алгоритм ошибся, так как за исключением одного нецензурного слова комментарий не содержит ярко выраженных негативно окрашенных слов. Помимо этого то единственное нецензурное слово, которое здесь использовано, употреблено в довольно специфической форме и, вероятно, больше нигде не встретилось, поэтому алгоритм не имел информации о том, токсичное ли это слово.

В четвертом примере алгоритм нашел слово, которое, как и во втором примере, довольно часто встречалось в токсичных комментариях, и поэтому имело больший вес в сторону токсичности, чем все остальные. На самом деле, в данном примере, хоть он и определен, как не токсичный, можно было бы увидеть и токсичный подтекст. Так что мы должны учитывать веротяность, что некоторые не токсичные комментарии могли оказаться токсичными, но в ходе их классификации произошла ошибка.

3.8.2 Вывод

Таким образом, наш классификатор показал **accuracy** = 0.861, что является довольно неплохим результатом. Проанализировав ошибки, с которыми он столкнулся, мы выяснили, что информация о контексте действительно ценна, а в используемых данных могли встретиться ошибочно определенные комментарии. Кроме того, комментарии на иностранном языке почти всегда не классифицировались из-за недостатка информации, а судя по таблице ошибок(рис.36) можно предположить, что их было около 170.

4 Вывод

При выполнении серии экспериментов были изучены свойства стохастического градиентного спуска и обычного градиентного спуска, установлены их схожие черты, а также отличия. Также мы изучили различные способы предобработки текстов и способы их хранения. Кроме того, был установлен наилучший алгоритм - стохастический градиентный спуск и подобраны оптимальные параметры для его использования, а также выбраны удобное представление данных и подходящая предобработка. При его использовании удалось получить точность, равную 86,1%.

5 Приложение

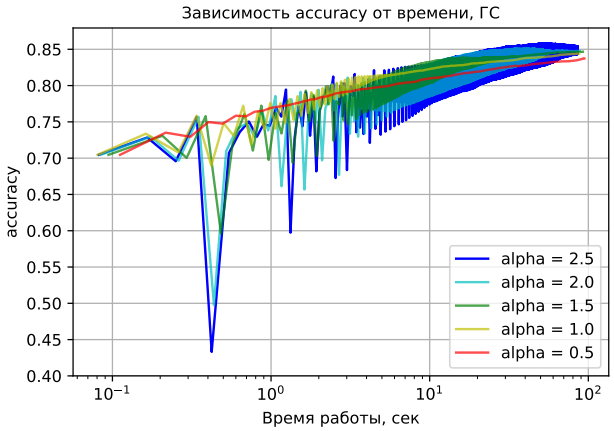

Рис. 37: Градиентный спуск

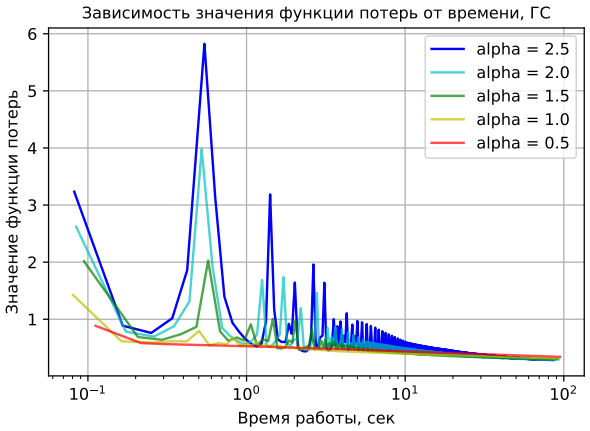

Рис. 38: Градиентный спуск

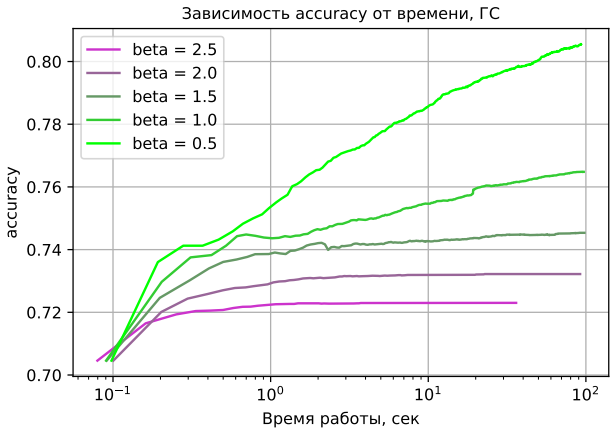

Рис. 39: Градиентный спуск

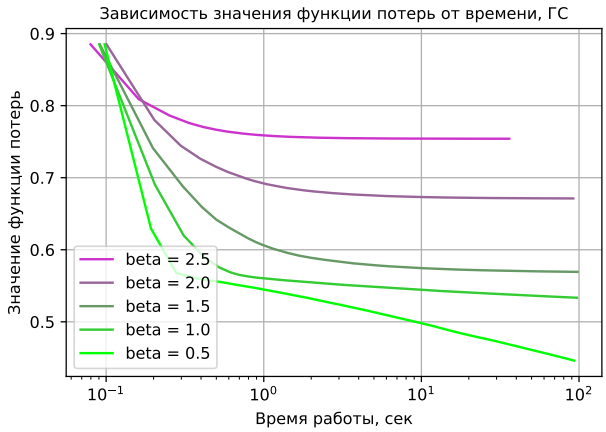

Рис. 40: Градиентный спуск

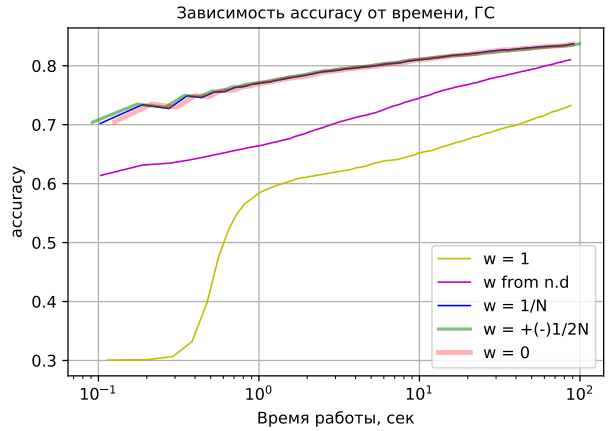

Рис. 41: Градиентный спуск

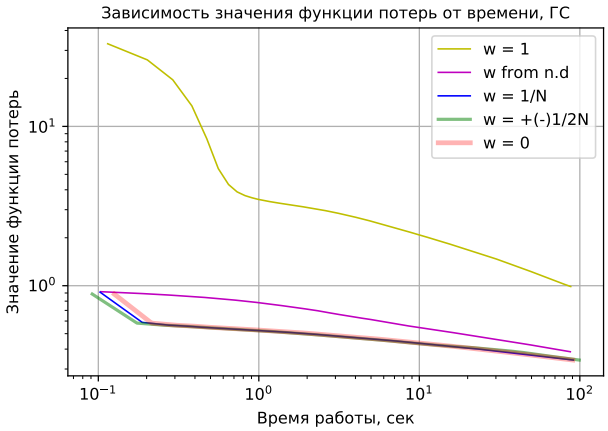

Рис. 42: Градиентный спуск

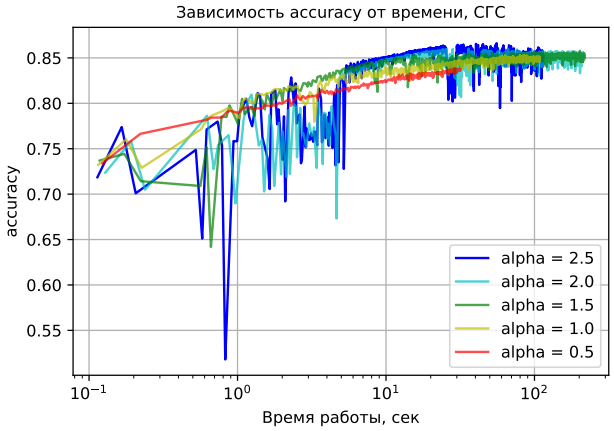

Рис. 43: Стохастический градиентный спуск

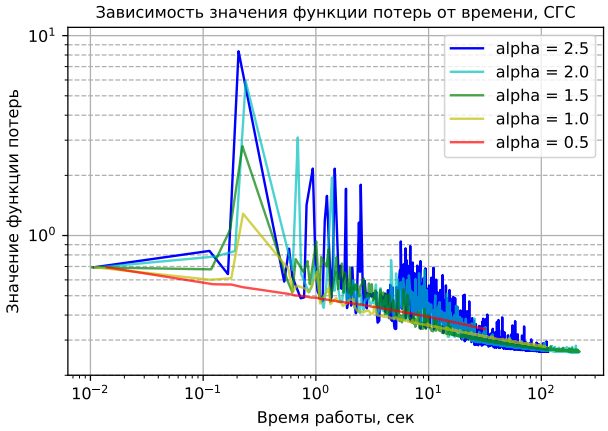

Рис. 44: Стохастический градиентный спуск

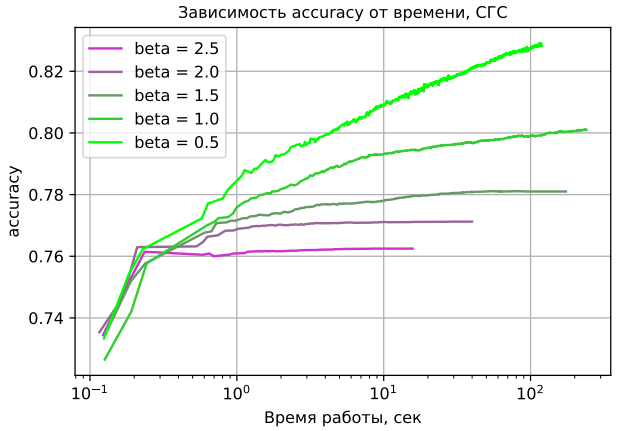

Рис. 45: Стохастический градиентный спуск

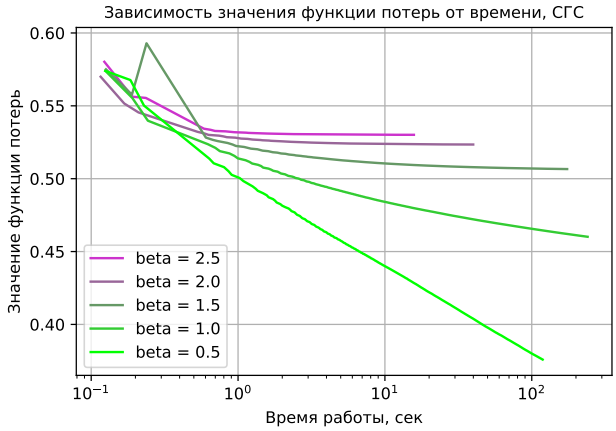

Рис. 46: Стохастический градиентный спуск

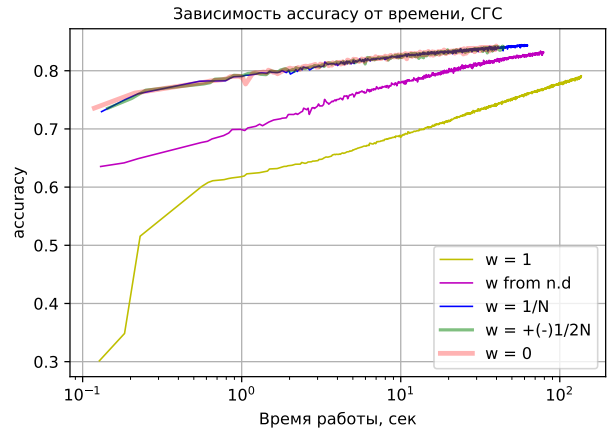

Рис. 47: Стохастический градиентный спуск

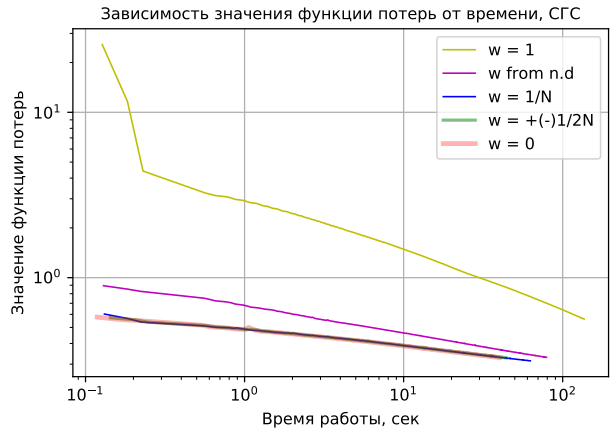

Рис. 48: Стохастический градиентный спуск

Таблица времени работы без предобработки(СГС)

	0.1	0.3	0.5
0.9	95.23	36.18	111.22
1.3	114.09	50.01	41.39

Рис. 49:

Таблица времени работы с предобработкой(СГС)

	0.1	0.3	0.5
0.9	90.37	124.07	175.90
1.3	66.40	138.60	177.64

Рис. 50:

Таблица точности без предобработки(ГС)

	0.1	0.3	0.5
0.9	0.843	0.831	0.817
1.3	0.846	0.834	0.822

Рис. 51:

Таблица точности с предобработкой(ГС)

	0.1	0.3	0.5
0.9	0.846	0.840	0.832
1.3	0.849	0.842	0.835

Рис. 52:

Таблица точности(ГС, BagOfWords)

	0.1	0.3	0.5
0.9	0.843	0.831	0.817
1.3	0.846	0.834	0.822

Рис. 53:

Таблица точности(ГС, Tfidf)

	0.1	0.3	0.5
0.9	0.805	0.766	0.737
1.3	0.813	0.779	0.746

Рис. 54:

Таблица времени работы(СГС, BagOfWords)

	0.1	0.3	0.5
0.9	95.23	36.18	111.22
1.3	114.09	50.01	41.39

Рис. 55:

Таблица времени работы(СГС, Tfidf)

	0.1	0.3	0.5
0.9	102.77	191.03	308.78
1.3	49.89	122.13	354.79

Рис. 56: