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

Отчет по градиентным методам обучения линейных моделей

Выполнил:

Студент 3 курса
Н. А. Панин

Москва
2022

1 Введение

В задачах машинного обучения часто возникает необходимость минимизировать гладкие выпуклые функционалы. Задача нахождения минимума у таких функционалов хорошо решается градиентными методами.

В данной работе рассматривалась логистическая регрессия (*Logistic regression*) - линейный классификатор, главным преимуществом которого является корректное определение вероятности принадлежности объекта определенному классу. В логистической регресии необходимо минимизировать эмпирический риск с логистической функцией потерь. На примере этого функционала была решена задача минимизации эмпирического риска для данных, содержащих в качестве признака комментарии из википедии, а в качестве целевой переменной метку "токсичный"\"не токсичный". Рассматривались такие градиентные методы как: *градиентный спуск* и *стохастический градиентный спуск*. Исследовались оптимальные параметры данных методов для минимизации функции потерь. Оценивалось влияние предобработки текста с помощью лемматизации и выкидывания стоп-слов на точность предсказания, время работы, размер признакового пространства. Также были рассмотрены два метода векторизации: BagOfWords и Tf-Idf. В конце были проанализированы и указаны общие черты объектов, на которых была допущена ошибка.

2 Теоретическая часть

Рассмотрим задачу бинарной логистической регрессии. Пусть дана обучающая выборка $X = (x_i, y_i)_{i=1}^N$, где $x_i \in \mathbb{R}^D$, $y_i \in \mathbb{Y} = \{1, -1\}$, $\omega \in \mathbb{R}^D$ – вектор весов.

1. Функция потерь будет иметь следующий вид:

$$Q(X, y, \omega) = \frac{1}{N} \sum_{i=1}^{N} \ln\big(1 + \exp(-y_i \langle x_i, \omega \rangle)\big) \to \min_{\omega}$$

Найдем дифференциал функции потерь по ω:

$$d_w Q(X, y, \omega) = -\frac{1}{N} \sum_{i=1}^{N} \frac{\exp(-y_i \langle x_i, \omega \rangle) y_i x_i^T d\omega}{1 + \exp(-y_i \langle x_i, \omega \rangle)}$$

$$\Longrightarrow \bigtriangledown_w Q(X, y, \omega) = \Big\{\sigma(z) = \frac{1}{1 + e^{-z}}\Big\} = -\frac{1}{N} \sum_{i=1}^{N} x_i y_i \sigma(-y_i \langle x_i, w \rangle),$$

где $\sigma(z)$ - сигмоида

2. Рассмотрим задачу мультиноминальной логистической регрессии, т.е. $\mathbb{Y} = \{1, 2, \ldots, K\}$. Тогда функция потерь будет иметь следующий вид:

$$Q(X, y, \omega) = -\frac{1}{N} \sum_{i=1}^{N} \ln\Big(\frac{\exp(\langle x_i, \omega_{y_i} \rangle)}{\sum_{k=1}^{K} \exp(\langle x_i, \omega_k \rangle)}\Big) =$$

$$= \frac{1}{N} \sum_{i=1}^{N} \left[\ln\Big(\sum_{k=1}^{K} \exp(\langle x_i, \omega_k \rangle)\Big) - \langle x_i, \omega_{y_i} \rangle\right] \to \min_{\omega_1, \omega 2, \ldots, \omega_K \in \mathbb{R}^D}$$

Градиентом в данном случае будет матрица размера $D \times K$:

$$dQ_{w_j}(X, y, \omega) = \frac{1}{N} \sum_{i=1}^{N} \left[\frac{\sum_{k=1}^{K} \exp(\langle x_i, \omega_k \rangle) d\langle x_i, \omega_k \rangle}{\sum_{k=1}^{K} \exp(\langle x_i, \omega_k \rangle)} - d\langle x_i, \omega_{y_i} \rangle\right] =$$

$$= \frac{1}{N} \sum_{i=1}^{N} \left[\frac{\exp(\langle x_i, \omega_j \rangle) x_i^T d\omega_j}{\sum_{k=1}^{K} \exp(\langle x_i, \omega_k \rangle)} - x_i^T [y_i = j] dw_j\right]$$

$$\Longrightarrow \bigtriangledown_{w_j} Q(X, y, \omega) = \frac{1}{N} \sum_{i=1}^{N} \left[\frac{\exp(\langle x_i, \omega_j \rangle) x_i}{\sum_{k=1}^{K} \exp(\langle x_i, \omega_k \rangle)} - x_i [y_i = j]\right]$$

$\bigtriangledown_{w_j} Q(X, y, \omega)$ - это j-й столбец этой матрицы.

3. Покажем связь между бинарной логистической регрессией и мультиноминальной логистической регресии, а именно докажем, что бинарная логистическая регрессия является частным случаем мультиноминальной. Действительно, для бинарной логистической регрессии:

$$\mathbb{P}(y = 1|x) = \sigma(\langle w, x\rangle)$$

Для мальтиноминальной логистической регрессии при $K = 2$:

$$\mathbb{P}(y = 1|x) = \frac{\exp(\langle \omega_1, x\rangle)}{\exp(\langle \omega_1, x\rangle) + \exp(\langle \omega_{-1}, x\rangle)} = \frac{1}{1 + \exp(\langle \omega_{-1} - \omega_1, x\rangle)} =$$

$$= \sigma(\langle \omega_1 - \omega_{-1}, x\rangle)$$

Так как задача нахождения оптимального решения в случае бинарной и мультиноминальной логистической регрессии сводится к максимизации функции правдоподобия, а вероятности равны, если положить $\omega = \omega_1 - \omega_{-1}$, то отсюда следует эквивалентность данных методов при $K = 2$.

3 Список экспериментов

Перед проведением экспериментов была проведена предобработка текстов: приводились слова к нижнему регистру, не буквы и не цифры удалялись и заменялись на пробелы.

Обучающая выборка была перемешена и разделена на валидационную и обучающую в отношении 1 : 4.

Значения некоторых параметров не изменялись почти во всех экспериментах. Если эти значения будут меняться, то это будет отдельно оговорено. Перечислим значения этих параметров по умолчанию:

- Максимальное количество итераций(эпох в случае стохастического градиентного спуска)[*max_iters*][1]: 100
- Точность[*tolerance*]: 10^{-10}
- Коэффициент l_2-регуляризации: 0.1
- Размер батча(подвыборки для подсчета градиента в SGD)[*batch_size*]: 10000

[1]Далее для сокращения записи в случае *градиентного спуска* будем писать **GD**, а в случае *стохастического градиентного спуска* **SGD**

- Частота обновлений "истории"(время, accuracy и значение функции потерь) SGD: 1 эпоха

Рассмотрим формулу итерационного нахождения минимума:

$$\omega_{k+1} = \omega_k - \frac{1}{L}\eta_k \sum_{i=1}^{L} \bigtriangledown_\omega Q(X, y, \omega_k), \tag{1}$$

где L - это либо *batch_size*, либо размер выборки X.

В качестве темпа обучения(*learning rate*) бралась

$$\eta_k = \frac{\alpha}{k^\beta} \tag{2}$$

3.1 Эксперимент 0. Анализ аналитической и разностной формулы градиента для логистической регрессии.

3.1.1 Дизайн эксперимента

В ходе данного эксперимента производилась разностная проверка градиента, а именно, использовалось следующее:

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

где $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)$; i – индекс, в котором стоит единица, а $\varepsilon \sim \sqrt{\varepsilon_{mach}}$, где ε_{mach} - машинная точность ($\approx 10^{-16}$). Причем в нашей задаче в качестве $f(\omega)$ будет $Q(X, y, \omega)$. В качестве параметров(X, y) и переменной(w) брались следующие значения:

- $X \sim \mathcal{U}(-100, 100)$, $X \in \mathbb{R}^{m \times n}$, где m и n случайные целые числа с равномерным распределением от 3 до 7.
- $y \in \mathbb{R}^{m \times 1}$ - случайный вектор из -1 и 1
- $w \sim \mathcal{U}(-5, 5)$, $w \in \mathbb{R}^{n \times 1}$

Вектор ω случайным образом брался 10 раз, считался по нему теоретический градиент и численный. Считалась средняя абсолютная ошибка по координатам вектора ω, а далее считалась средняя ошибка по всем 10 измерениям.

3.1.2 Результаты эксперимента

В ходе эксперимента была получена следующая средняя ошибка: 7.7×10^{-7}

3.1.3 Выводы эксперимента

Таким образом, во-первых, была проверена правильность посчитанной аналитической формулы градиента, поскольку средняя ошибка очень мала. Во-вторых, было эвристически показано, что при условии правильности аналитической формулы градиента разностная формула хорошо приближает значение градиента в точке(в нашем примере с точностью до 5-го порядка)

3.2 Эксперимент 1. Анализ оптимальных параметров для GD.

3.2.1 Дизайн эксперимента

В ходе данного эксперимента исследовались параметры для GD:

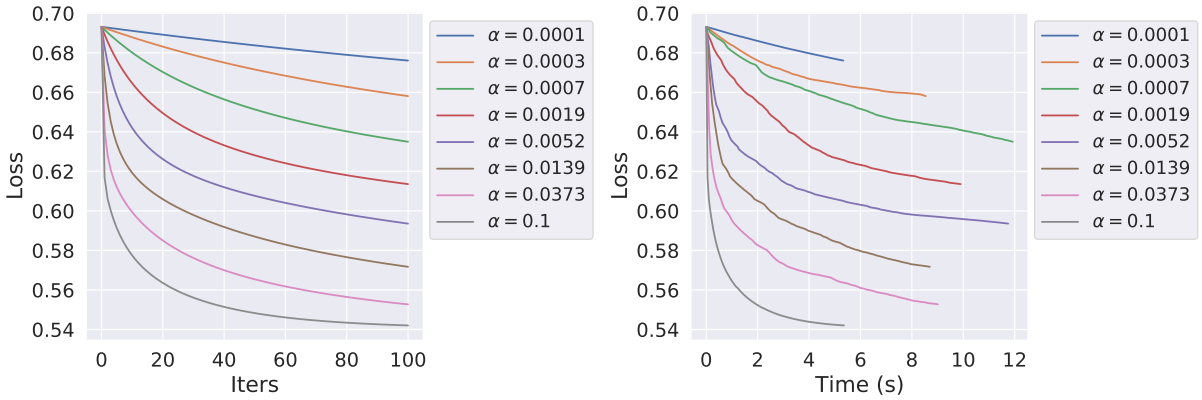
- α - размер шага в 2 (по логарифмической сетке)
- β - размер шага в 2 (по логарифмической сетке)
- ω_0 - начальное приближение в 1

В данном эксперименте при выборе оптимального параметра остальные фиксировались. Выбор наилучшего параметра определялся из значений *accuracy* на валидационной выборке. При поиске следующих оптимальных параметров в качестве значений уже обработанных параметров брались найденные для них наилучшие параметры. Таким образом, в ходе эксперимента изучался не только вопрос устойчивой оптимальной сходимости функции потерь к минимуму, но также жадно подбирались наилучшие параметры, что в последующем будет использовано в последнем эксперименте.

3.2.2 Результаты эксперимента

1. **Выбор α**

   Фиксируем $\beta = 1$ и $\omega_0 = 0 \in \mathbb{R}^D$ и по логарифмической сетке рассмотрим различные параметры α (см. рис.1). Все графики зависи-



(a) (b)

Рис. 1

мости от времени и итераций очень похожи друг на друга, поэтому в зависимости от потребностей в дальнейшем будут использоваться как графики от итераций так и графики от времени, а остальные будут помещены в **Приложении А**. Из рис.1 видно, что чем меньше α тем медленнее сходится к минимуму итерационный процесс, ему необходимо больше шагов итераций, так как темп обучения слишком мал. Впрочем, и слишком большие значения α не способствуют хорошей оптимизации (см. рис. 2). В результате выбора больших $\alpha \gtrsim 1.0$

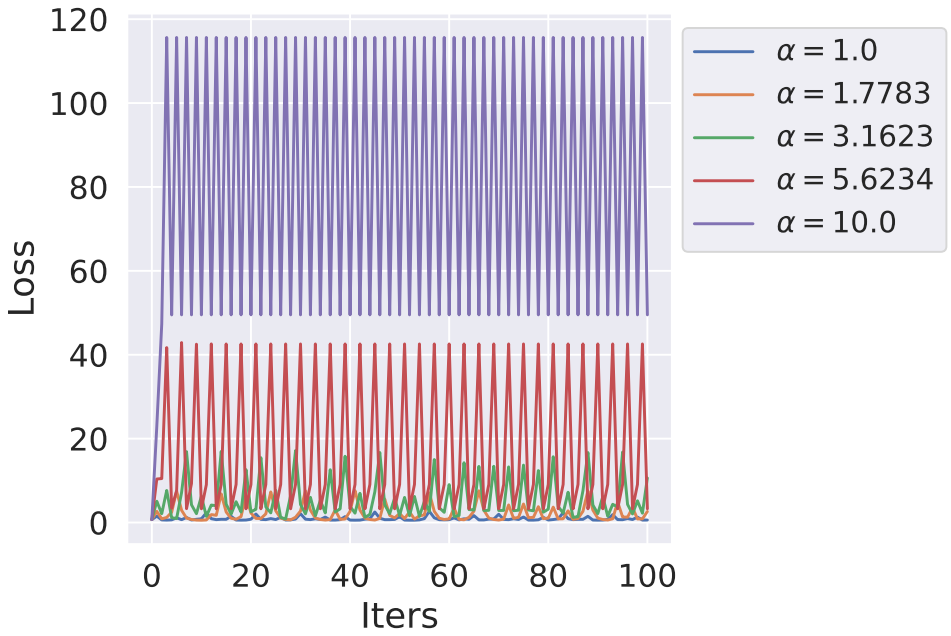
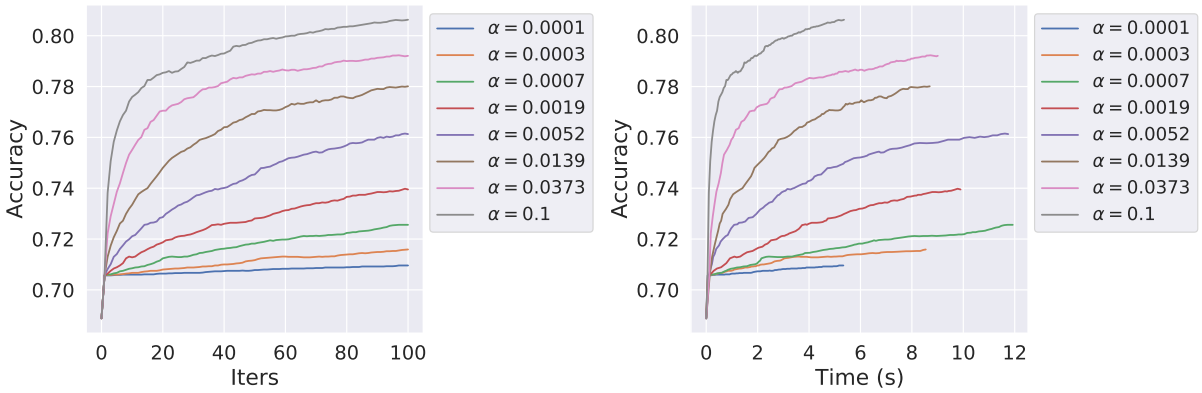


Рис. 2

уже наблюдается сильная осцилляция. При дальнейшем увеличение

градиентный спуск на каждой итерации будет осциллировать около точки минимума(причем из графика видно, что при $\alpha = 10$ осцилляция уже происходит в области другой точки минимума). Отрицательные значения α бессмысленны, так как мы решаем задачу минимизации, а значит, нам нужен антиградиет, то есть градиент со знаком минус. Как видим из рис.3 с увеличением α (при достаточно



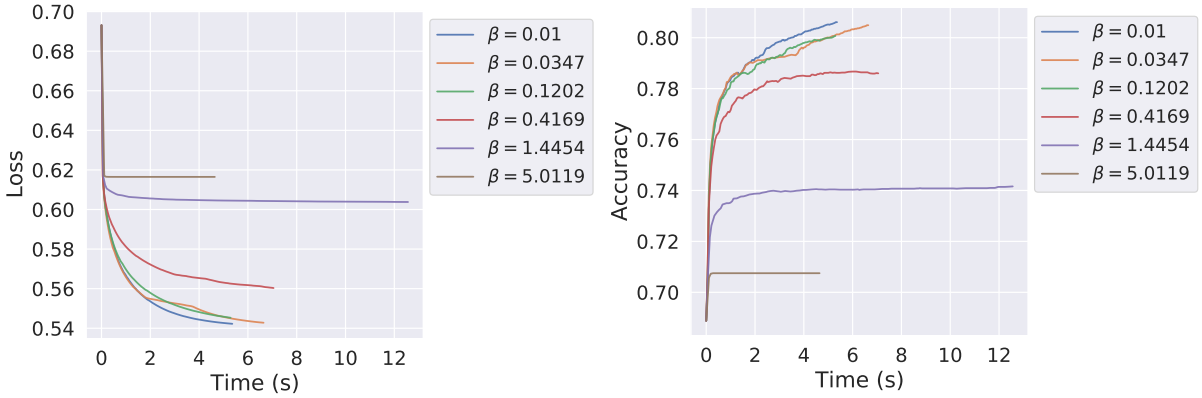
(a) (b)

Рис. 3

малых α) быстрее достигается лучший accuracy.

2. **Выбор** β

Теперь фиксируем $\alpha = 0.1$. При этом ω_0 оставим той же. Будем перебирать по логарифмической сетке параметры β



(a) (b)

Рис. 4

Из графика (рис.4) видно, что с уменьшением β оптимизационная задача быстрее сходится к минимуму. При слишком больших $\beta \gtrapprox 5$

алгоритм очень быстро закончит работать, поскольку из-за маленьких темпов обучения ω_k(см. (1)) будет не значительно меняться. С учетом гладкости функционала (см. Теоретическую часть) и его значения будут мало изменяться, следовательно условие останова цикла ($|Q(w_{k+1}) - Q(w_k)| < tolerance$) будет быстро достигнуто и функция плохо решит оптимизационную задачу.

Из рис.4 следуют те же выводы, что и в случае с α: с уменьшением β оптимальная точность быстрее достигается

3. **Выбор** ω_0

Фиксируем $\alpha = 0.1$, $\beta = 0.01$. Переберем ω_0(см рис.5).

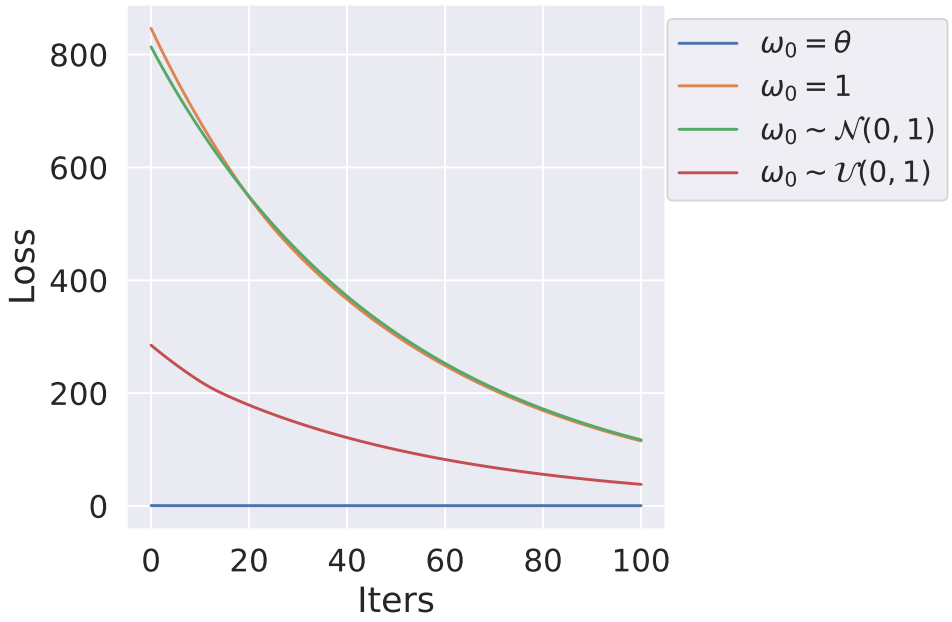


Рис. 5

Некоторые пояснения:

- $\mathcal{N}$ - нормальное распределение
- $\mathcal{U}$ - равномерное распределение

Из графика (см. рис.5) видно, что с увеличением нормы ω_0 значение функции потерь принимает большие значения. Лучшим вариантом будет взять $\omega_0 = 0$, тогда будет тратиться меньшее количество итераций на минимизацию, так как иначе сначала уйдет большое количество итераций на уменьшение больших значений функционала, а потом проблема возникнет с шагом (см. 2), так как он уже будет мал

и значения функционала будут незначительно изменяться.
Теперь посмотрим на точность при различных ω_0 (см. рис. 6)

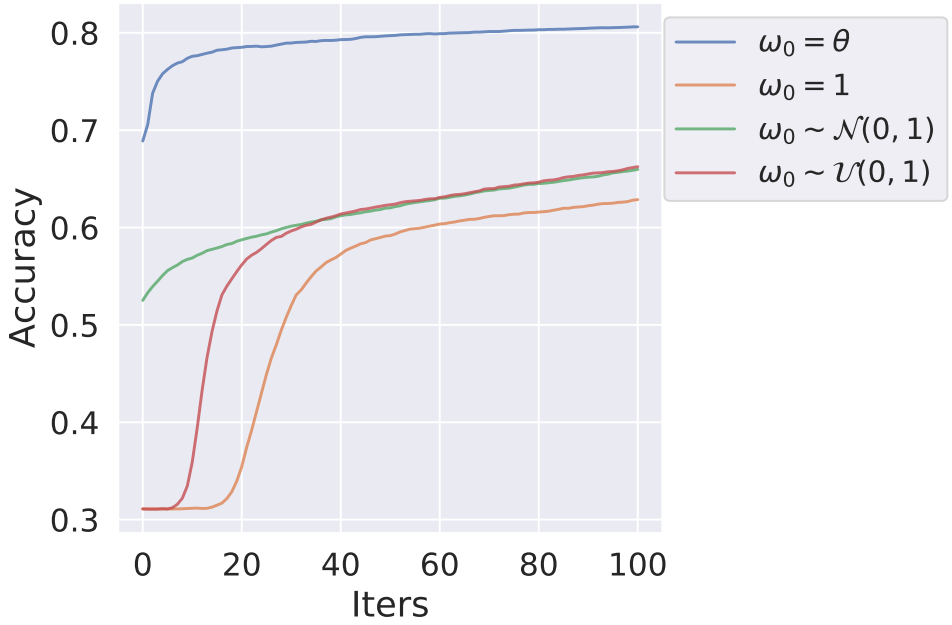


Рис. 6

Точность также показывает правильность выбора начального приближения равного нулю.

3.2.3 Выводы эксперимента

Для градиентного спуска оптимальным будет следующий выбор параметров:

- $\alpha \lessapprox 1.0$
- $\beta \lessapprox 0.15$
- $w = 0,\ 0 \in \mathbb{R}^D$

3.3 Эксперимент 2. Анализ оптимальных параметров для SGD.

3.3.1 Дизайн эксперимента

В ходе данного эксперимента исследовались параметры для SGD. Дизайн эксперимента полностью повторяет предыдущий эксперимент за ис-

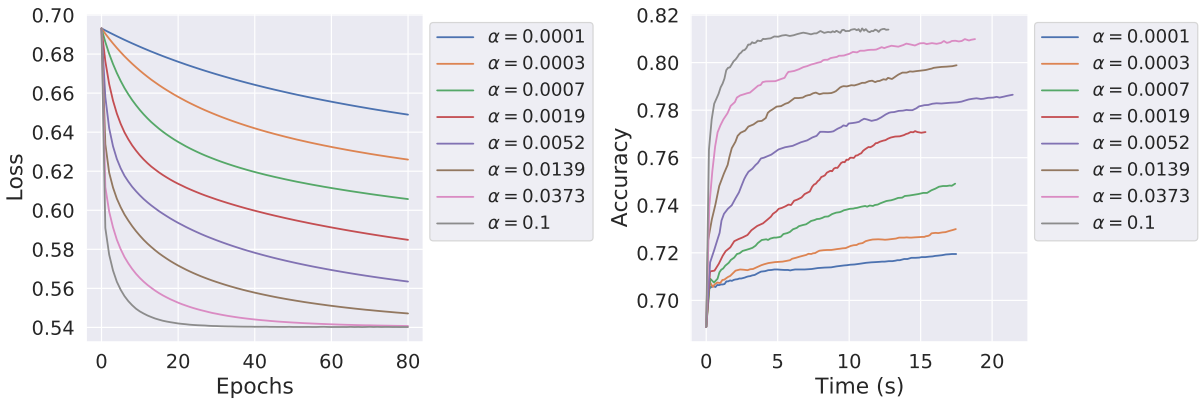
ключением того факта, что к параметрам из предыдущего эксперимента добавился размер батча ($batch\_size$).

3.3.2 Результаты эксперимента

Сразу же заметим, что зависимость графиков от времени и от эпох очень похожи, поэтому все графики будут находиться в **Приложении Б**, а здесь же используются только необходимые.

1. **Выбор** α

   Фиксируем $\beta = 1$, $\omega_0 = 0 \in \mathbb{R}^D$, $batch\_size = 10000$ и по логарифмической сетке рассмотрим различные параметры α (см. рис.7): Из



(a) (b)

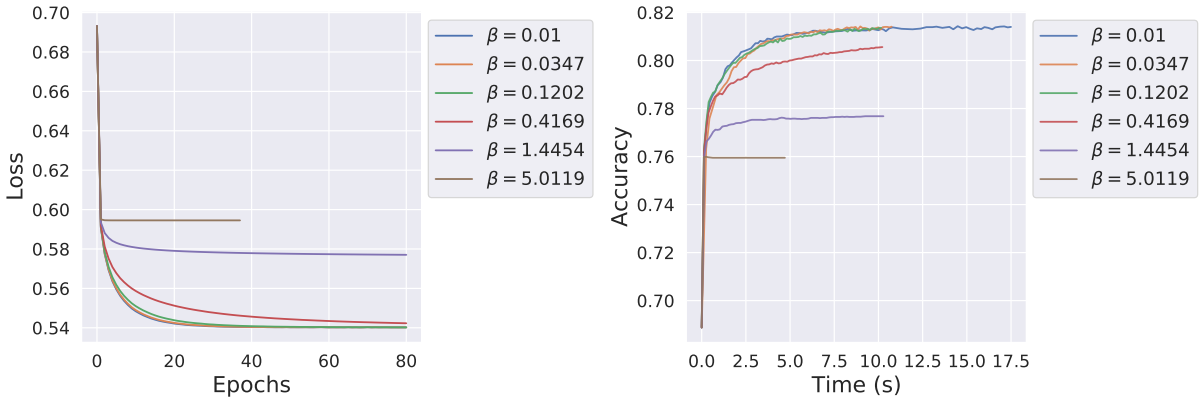
Рис. 7

   графика (см. рис.7) следуют точно такие же выводы как и в случае градиентного спуска, то есть при увеличении α (не больше 1) функция потерь быстрее стремиться к минимуму. При больших α уже видна осцилляция, а при дальнейшем увеличении задача минимизации уже не разрешима (см. **Приложение Б**).

   Посмотрим на accuracy. Как видно из рис. 7 на валидационной выборке точность наилучшая при $\alpha = 0.1$, при этом время работы при различных значениях данного параметра почти одинаково

2. **Выбор** β

   Фиксируем $\alpha = 0.1$, $\omega_0 = 0 \in \mathbb{R}^D$, $batch\_size = 10000$ и по логарифмической сетке рассмотрим различные параметры β (см. рис.8): Видно (см. рис.8), что тут такая же зависимость как и в случае гра-



(a) (b)

Рис. 8

диентного спуска, то есть при уменьшении β функция потерь быстрее стремиться к минимуму. При больших β быстро выходит на плато и соответственно за короткое время достигает условия останова, о котором упомяналось выше.

Исследуем accuracy. Наилучшим оказался $\beta = 0.0347$ ($accuracy = 0.8141$), при этом при $\beta \lessapprox 0.15$ $accuracy$ тоже весьма хороший ≈ 0.813.

3. **Выбор ω_0**

Фиксируем $\alpha = 0.1$, $\beta = 0.0347$, $batch\_size = 10000$ и рассмотрим различные параметры ω_0 (см. рис.9, рис.10):

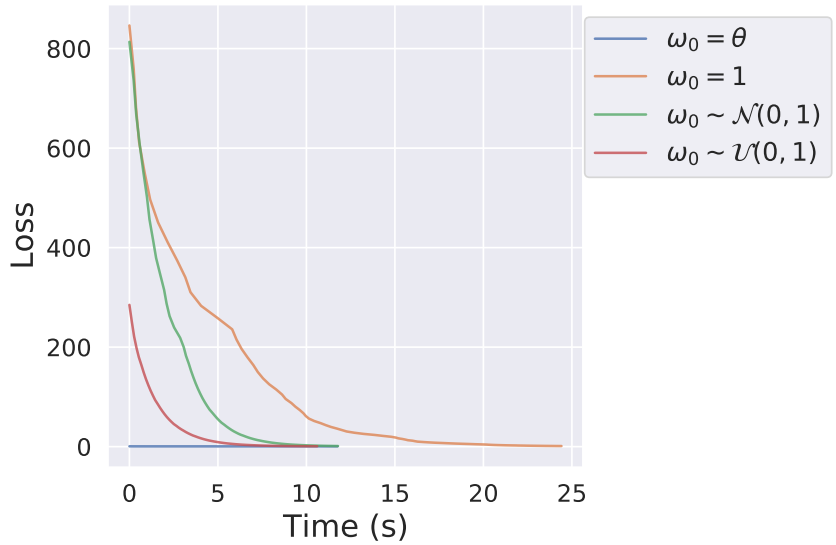


Рис. 9

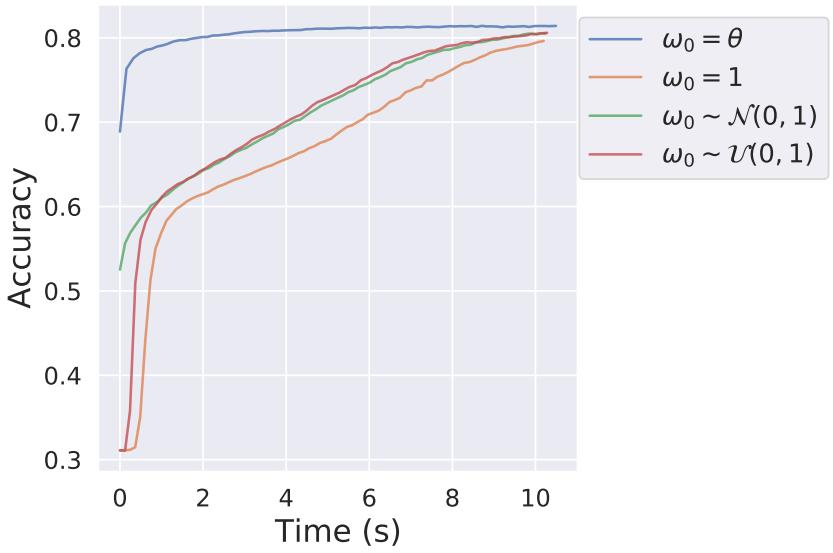


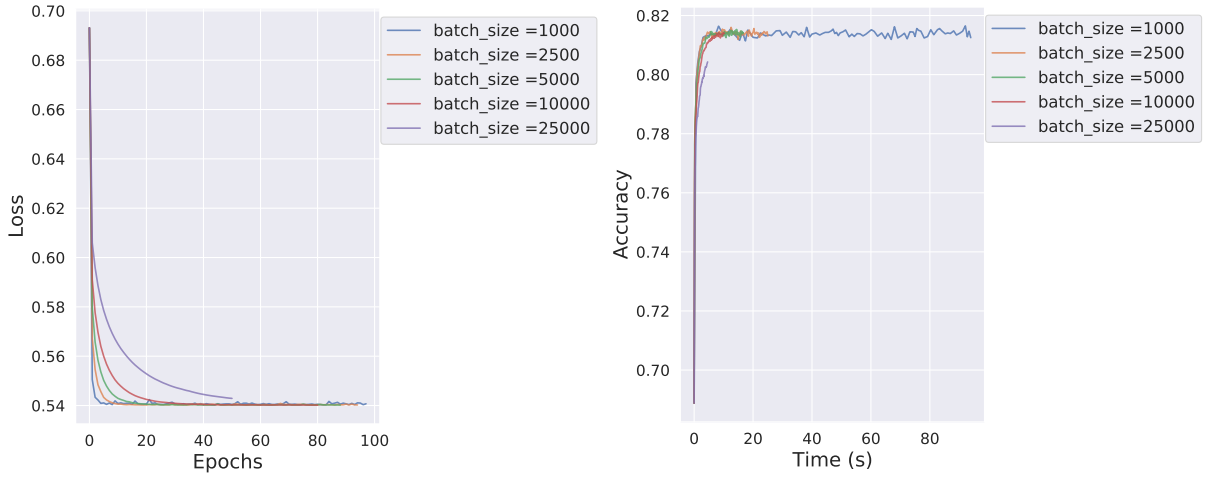
Рис. 10

По-прежнему лучшим начальным приближением является $\omega_0 = 0$. Из необычного несложно заметить, что в отличие от эксперимента 1[2] в этом *accuracy* для всех рассмотренных случаев значительно ближе к наилучшей оценке минимума для $\omega_0 = 0$ и достигается за меньшее количество времени чем для GD.

4. **Выбор** *batch_size*

Фиксируем $\alpha = 0.1$, $\beta = 0.0347$, $\omega_0 = 0$ и рассмотрим различные параметры batch_size (см. рис.11): Проанализировав рис.11, можно заметить что чем меньше размер батча тем меньше эпох, а значит, меньше потраченного времени нужно для достаточно неплохих оценок минимума функционала. Также видно, что с уменьшением батча увеличивается осцилляция. Хотя наилучшая точность на валидационной выборке была достигнута с $batch\_size = 1000$ ($accuracy = 0.8145$), в дальнейшем будет использоваться $batch\_size = 10000$ ($accuracy = 0.8141$), поскольку он оптимален по нескольким параметрам: время (в $\approx$8 раз быстрее работает чем с $batch\_size = 1000$), *accuracy* и устойчивость(меньше осцилляций, а значит при одних и тех же значениях метод будет получать более устойчивые оценки точности, чем с меньшими значениями размера)

[2]В эксперименте 1 *accuracy* для $\omega_0 \neq 0$ равен ≈ 0.65



(a) (b)

Рис. 11

3.3.3 Выводы эксперимента

Для стохастического градиентного спуска оптимальным будет следующий выбор параметров:

- $\alpha \lessapprox 1.0$
- $\beta \lessapprox 0.15$
- $w = 0,\ 0 \in \mathbb{R}^D$
- $batch\_size \leq 10000$

3.4 Эксперимент 3. Сравнение SGD и GD.

3.4.1 Дизайн эксперимента

Для сравнения двух методов бралось множество оптимальных параметров, подобранных в предыдущих экспериментах, а именно:

- Для GD:
  1. $\alpha = 0.1$
  2. $\beta = 0.01$
  3. $\omega_0 = 0 \in \mathbb{R}^D$
- Для SGD:
  1. $\alpha = 0.1$

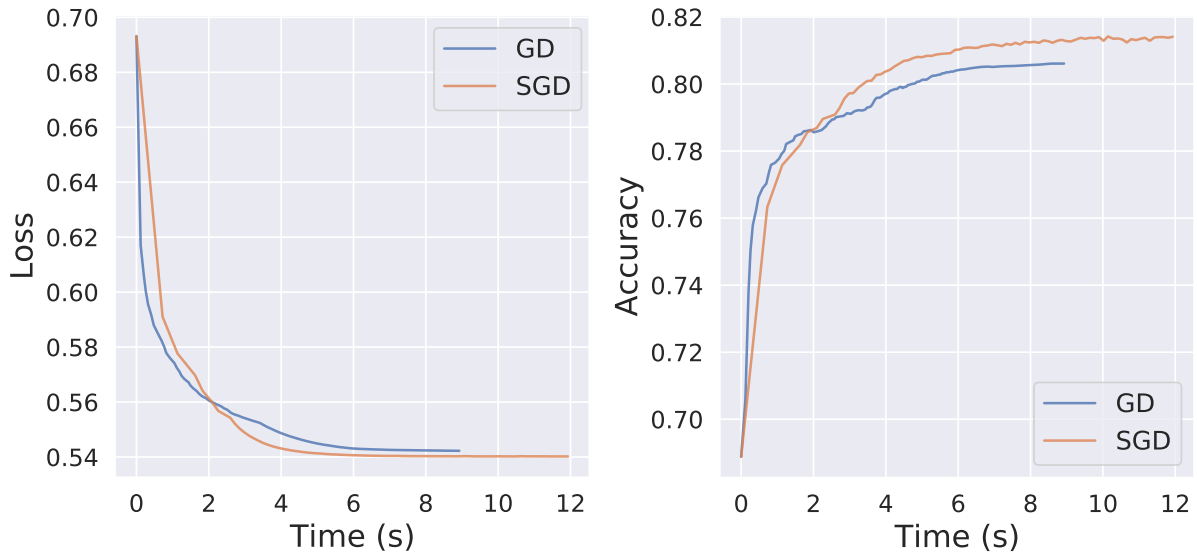
2. $\beta = 0.0347$

3. $\omega_0 = 0 \in \mathbb{R}^D$

4. $batch\_size = 10000$

Далее исследовались методы GD и SGD с данными параметрами по *accuracy*, времени и значению функции потерь.

3.4.2 Результаты эксперимента



(a) (b)

Рис. 12

Сравнение методов			
Название градиентного метода	Время (с)	Лосс	*accuracy*
GD	8.925143	0.542243	0.806100
SGD	11.928005	0.540191	0.814100

Из рис.12 и таблицы видно, что по точности и качеству оптимизации с точки зрения близости к минимуму, а не времени качественно лучше стохастический градиентный спуск. Вычислительная сложность для GD будет $O(N*D*max\_iters)$, а для SGD $O(N*D*max\_iters*\lceil\frac{N}{batch\_size}\rceil)$[3], поэтому не удивительно, что работать SGD будет дольше.

[3] max_iters для SGD это количество эпох

3.4.3 Выводы эксперимента

По качеству наилучшим методом оказался SGD, но по времени он проигрывает GD. При этом с уменьшением размера батча время работы SGD будет увеличиваться обратно пропорционально.

3.5 Эксперимент 4. Предобработка корпуса и ее влияние на качество модели.

3.5.1 Дизайн эксперимента

В данном эксперименте использовались те же параметры, что и в предыдущем:

- Для GD:
  1. $\alpha = 0.1$
  2. $\beta = 0.01$
  3. $\omega_0 = 0 \in \mathbb{R}^D$
- Для SGD:
  1. $\alpha = 0.1$
  2. $\beta = 0.0347$
  3. $\omega_0 = 0 \in \mathbb{R}^D$
  4. $batch\_size = 10000$

В этом эксперименте рассматривалось влияние на время работы градиентного метода, $accuracy$, и размерность признакового пространства D таких методов предобработки как:

- лемматизация(сведение слово к начальной форме, к лемме). Пример: "write wrote written" $\rightarrow$ "write write write"
- удаление стоп-слов (предлоги, частицы и другие шумовые слова). Пример: "The Moscow is the best city" $\rightarrow$ "Moscow best city"

Для определения влияния обработки рассматривались время, $accuracy$, размер признакового пространства по выборке до обработки и те же самые параметры по выборке после обработки. Сравнение велось как для SGD, так и для GD.

3.5.2 Результаты эксперимента

	Время (с)	Количество признаков	*accuracy*
GD без обработки	8.925143	16050	0.806100
GD с обработкой	5.081007	12964	0.820700
SGD без обработки	11.928005	16050	0.814100
SGD с обработкой	10.939810	12964	0.830700

Сразу заметим, что размерность признакового пространства сильно уменьшилась. Из-за уменьшения размерности пространства с учетом вычислительной сложности для GD и SGD, полученной выше в эксперименте 3, очевидно, уменьшится и время работы модели, что подтверждается значениями в таблице. Из таблицы также следует, что лемматизация и удаление стоп-слов способствует значительному улучшению точности: для GD на 7.52% меньше ошибок (на ≈ 145 меньше текстов-ошибок для валидационной выборки размера 10000), а для SGD на 8.92% меньше ошибок (на ≈ 165 меньше текстов-ошибок для валидационной выборки размера 10000).

3.5.3 Выводы эксперимента

Предобработка корпуса с помощью лемматизации и последующего удаления стоп-слов дает не только выигрыш в памяти, но и позволяет улучшить модель по времени и точности.

3.6 Эксперимент 5. Векторизация текстов. Выбор параметров min_df и max_df

3.6.1 Дизайн эксперимента

Заданы:

- Общие значения параметров, по которым идет перебор:
  1. $min\_df = 0.00001, 0.0001, 0.01, 0.1, 0.3$
  2. $max\_df = 0.1, 0.25, 0.35, 0.45, 0.55, 0.7$
- Для GD:
  1. $\alpha = 0.1$
  2. $\beta = 0.01$

3. $\omega_0 = 0 \in \mathbb{R}^D$

- Для SGD:

1. $\alpha = 0.1$
2. $\beta = 0.0347$
3. $\omega_0 = 0 \in \mathbb{R}^D$
4. $batch\_size = 10000$

В данном эксперименте анализируется влияние на точность, время работы, размер признакового пространства различных методов векторизации текстов, параметров min_df и max_df. В качестве методов векторизации будут рассмотрены два метода: **BagOfWords(BOW)**, основанный на подсчете количества повторений уникальных слов, и **Tf-Idf**, основанный на подсчете количества повторений уникальных слов с поправкой на частоту встречаемости во всей коллекции текстов. После изучения векторизации, будет рассмотрен min_df, а уже после того, как будет найден оптимальный min_df будет изучен вопрос об оптимальном max_df. Также стоит отметит, что в ходе этого эксперимента в качестве валидационной и обучающей выборки брались валидационноая и обучающую выборка, которая обсуждалась в самом начале раздела "Список экспериментов".

3.6.2 Результаты эксперимента

1. **BagOfWords и Tf-Idf**

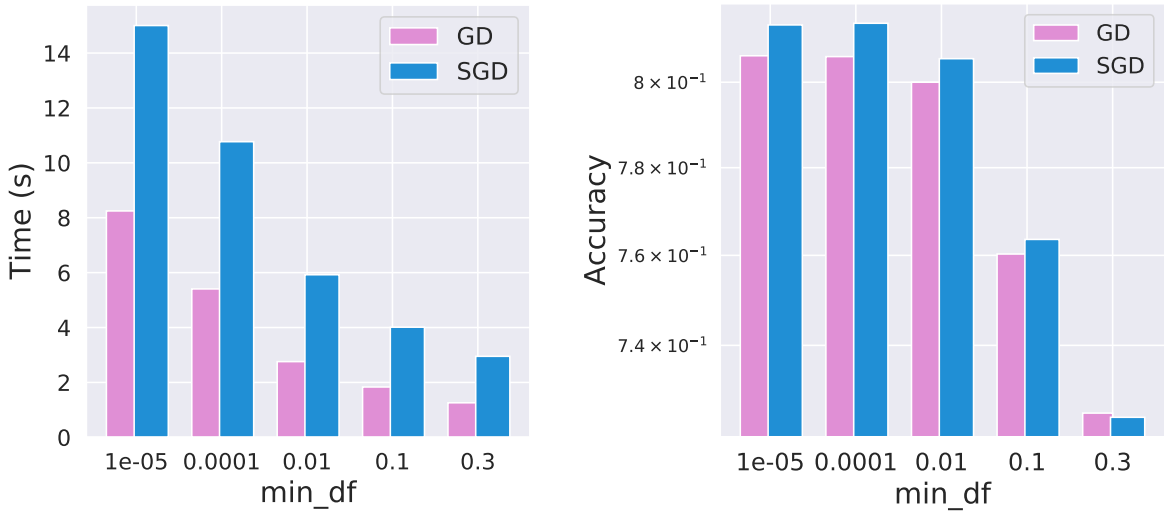
Метод	Вид векторизации	Время (с)	Количество признаков	*accuracy*
GD	BOW	7.759336	16050	0.806100
	Tf-Idf	6.127796	16050	0.721500
SGD	BOW	11.931548	16050	0.814100
	Tf-Idf	8.624678	16050	0.723900

Таблица выше была построена по предсказаниям на валидационной выборке.Заметим, что при данных методах векторизации размерность признакового пространства не изменяется. Время незначительно, но все же меньше в обоих случаях при использовании Tf-Idf. Что же касается *accuracy*, то, вероятно, в текстах часто можно встретить слова, например, "you", "like", которые, вообще говоря, присутствуют во многих текстах, а значит, будут иметь маленький

вес в Tf-Idf. При этом количество этих слов сильно возрастает в токсичных текстах для сравнения человека с чем-то, поэтому хотелось бы, чтобы мы это как-то учитывали в модели, но из-за малых весов эта информация не будет почти учитываться.
Примеры:

- "fuck you you stupid american"
- "you are such a dickhead you lesbian butch"

2. **min_df**



(a) (b)

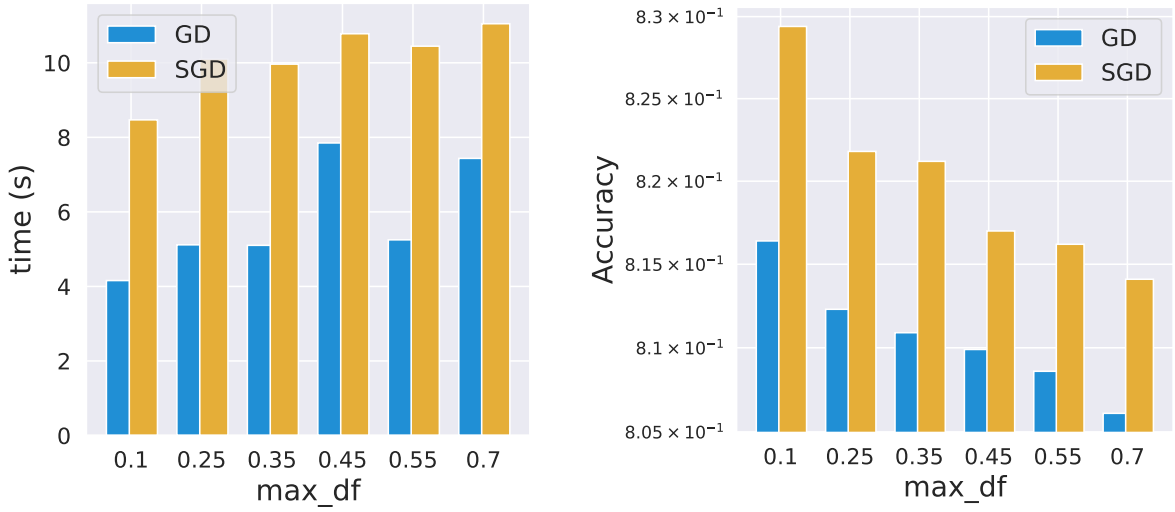
Рис. 13

Из рис.13 видно, что чем меньше значение min_df, тем GD и SGD больше тратят времени на минимизацию функционала $Q(w)$, это неудивительно, так как размер признакового пространства увеличивается (см. таблицу ниже)[4]. Также из графика зависимости $accuracy$ от min_df видно, что при $min\_df \lessapprox 0.0001$ точность примерно такая же как и при $min\_df = 0.0001$ (при $min\_df = 0.0001$ она даже выше чем при $min\_df = 10^{-5}$: 0.8141 против 0.8137), а при $min\_df \gtrapprox 0.0001$ уже начинается падение точности. Видимо, при больших min_df слишком много полезных признаков пропадает, так как признак(слово) выкидывается тогда, когда частота его встречаемости во всех текстах меньше заданного значения min_df.

[4]Оценки вычислительной сложности GD и SGD были даны выше в эксперименте 3

3. **max_df**

Фиксируем теперь оптимальный $min\_df = 0.0001$. Тут со временем ситуация такая же как и у min_df с увеличением max_df увеличивается признаковое пространство (см. таблицу ниже и рис.14), поэтому и время увеличивается. Также нетрудно заметить, что max_df с



(a) (b)

Рис. 14

увеличением своего значения лишь понижает точность алгоритма.

Таблица зависимости количества признаков от значений min_df и max_df

Параметр	Значение параметра	Количество признаков
min_df	0.00001	89368
	0.0001	16050
	0.01	568
	0.1	55
	0.3	11
max_df	0.1	15995
	0.25	16034
	0.35	16041
	0.45	16046
	0.55	16048
	0.7	16050

3.6.3 Выводы эксперимента

- Tf-Idf плохо использовать для задачи определения токсичности комментария, так как он может давать малые веса часто употребляемым словам, которые в токсичных текстах чаще обычного встречаются
- Из расмотренных параметров наилучшими являются $min\_df = 0.0001$ и $max\_df = 0.1$

3.7 Эксперимент 6. Лучший алгоритм. Анализ ошибок алгоритма.

3.7.1 Дизайн эксперимента

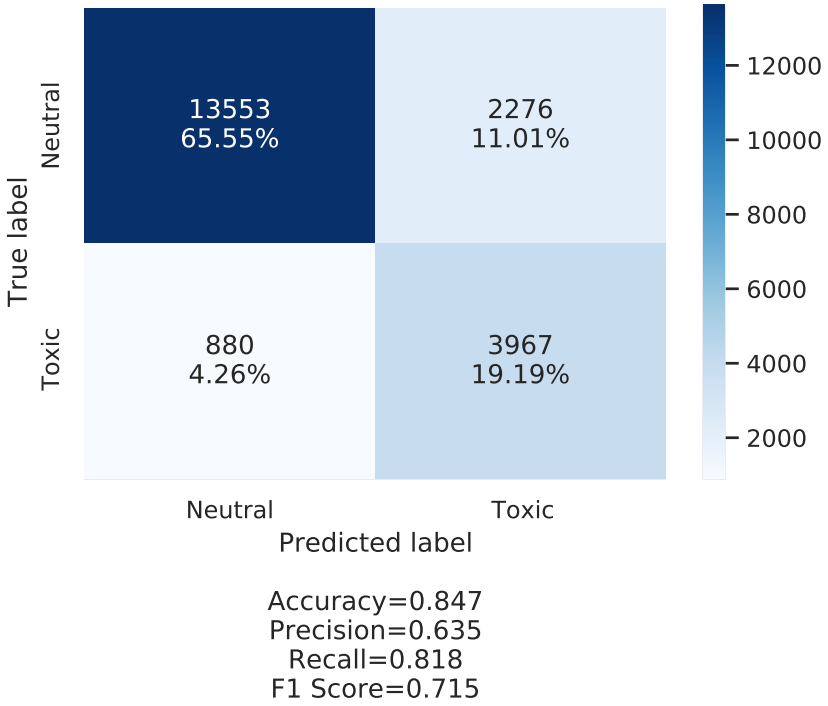
В ходе данного эксперимента брались лучшие параметры, подобранные на предыдущих шагах и лучший из разобранных алгоритмов (SGD):

1. $\alpha = 0.1$
2. $\beta = 0.0347$
3. $\omega_0 = 0 \in \mathbb{R}^D$
4. $batch\_size = 10000$
5. Текст подвергался лемматизации и выкидыванию стоп-слов
6. Использовалась модель векторизации BagOfWords
7. $min\_df = 0.0001$
8. $max\_df = 0.1$
9. $max\_iters = 1000$ (для увеличения точности)

В этом эксперименте в качестве обучающей и тестовой были взяты исходные данные, то есть никакого деления на валидационную выборку уже не происходило. После обучения на обучающей выборке была построена и проанализирована матрица ошибок предсказания на тестовых ответах. Также были разобраны типичные ошибки алгоритма на примерах.

3.7.2 Результаты эксперимента

Рис. 15: Матрица ошибок



Как видим из рис.15 следует, что модель хорошо классифицирует тексты, однако все же делает ошибки и чаще всего при классификации нейтральных текстов как токсичных. Рассмотри примеры текстов, на которых алгоритм чаще всего ошибается:

- *"i think the origin of sagging has his roots in that human stupidity has no limits", "alex think again mr obama is the us president what about putin and his background worse than an ass"*
  – эти предложения были классифицированы, как нейтральные, хотя не являются такими. Вероятно, это происходит из-за того, что в этих текстах большая часть слов по-отдельности (кроме "stupidity"и "ass") это вполне обычные слова, употребляемые и в нейтральных текстах.

- "she looks like a horse" – "horse" редко употребляется в качестве оскорбления, поэтому не верно классифицирован как нейтральный

- "f u c k e r c o m m i e" – из-за того, что есть пробельные слова это предложение будет представлено в виде вектора букв, а не сло-

ва. Понятно, что модель будет давать плохое предсказание, так как буквы не слова.

- *"we got snow here in boston too flying home tomorrow is going to suck", "racism in moby dick was herman melville a racist particularly chapter 42 of moby dick melville expounds on the terrifying aspects of the color white how white dominates other colors and how the white race dominates other races was moby dick the white whale a symbol of white superiority and pride i was shocked when i read this statement from chapter 42 pg 163 norton 1967 speaking on whiteness melville writes and though this preeminence in it applies to the human race itself giving the white man mastership over every dusky tribe"*
  - тексты неверно классифицированы как токсичные. Алгоритм, ошибается, поскольку, модель не знает контекст, и ошибочно воспринимает такие слова как "suck" и "dick" в "moby dick" как токсичные.

3.7.3 Выводы эксперимента

Таким образом, в основном алгоритм ошибается, поскольку не знает контекста, хотя бывает и в следствие наличия в текстовых данных данных, которые с формальной точки зрения не являются текстами (то есть непустой последовательностью, состоящей из слов, а не букв)

4 Общие выводы

В рамках проведенного исследования были достигнуты поставленные цели и решены сформулированные в начале исследования задачи. Особенно хотелось бы выделить следующие результаты:

- в рамках нашей задачи (задача определения токсичности комментария) лучшую обобщающую способность показал *стохастический градиентный спуск*

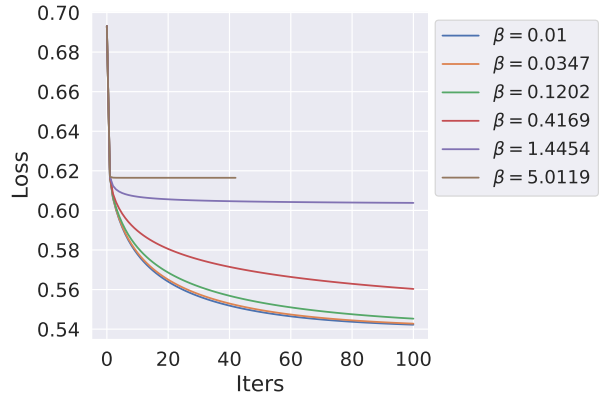
- Размерность признакового пространства влияет на время работы градиентых методов, поэтому предобработка текстов: лемматизация, удаление стоп-слов и пр. является хорошей практикой. В том числе благодаря такой предобработки можно существенно повысить обобщающую способность модели.

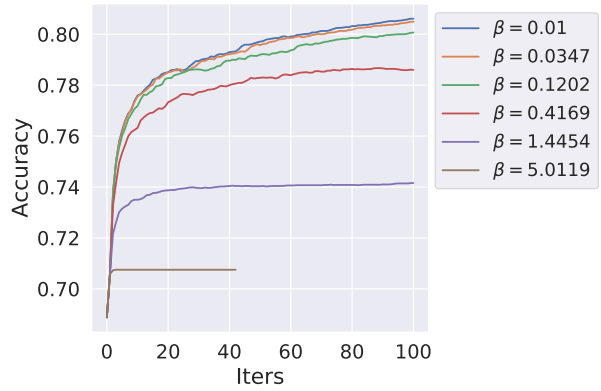
- *batch_size* влияет обратно пропорционально на время работы SGD.

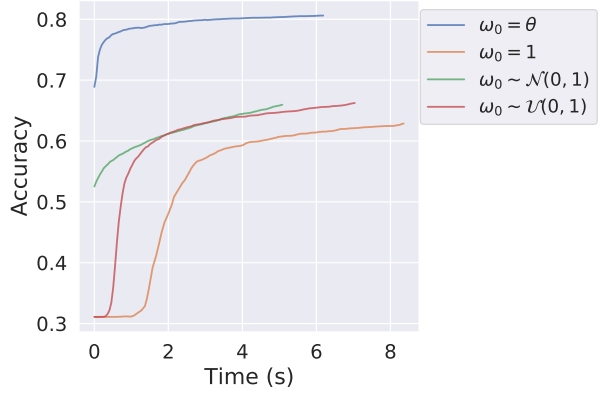
- выбор значения для *training rate* играет ключевую роль в задаче оптимизации: слишком маленькие значения являются причиной медленной сходимости, слишком большие значения осциллируют в окрестности минимума, так и не достигая его. Имеет смысл эвристически подбирать данный параметр.

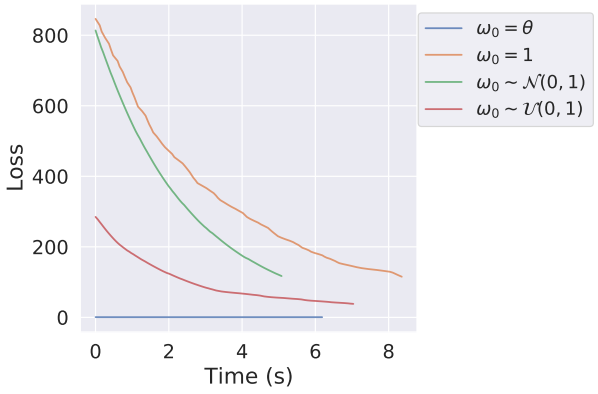
- Tf-Idf не идеален, были рассмотрены случаи, когда BagOfWords превосходит его (например, в данной задаче так и было)

5 Приложение A. Градиентный спуск.

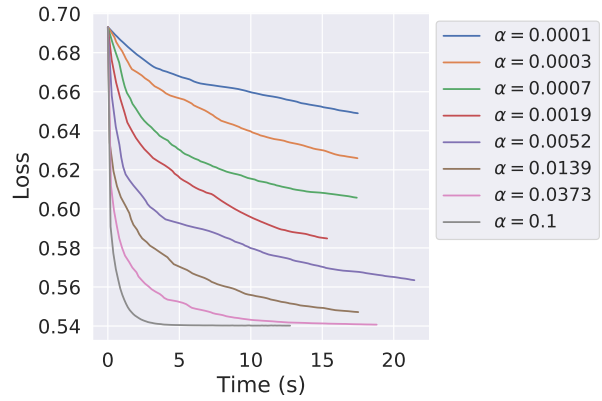


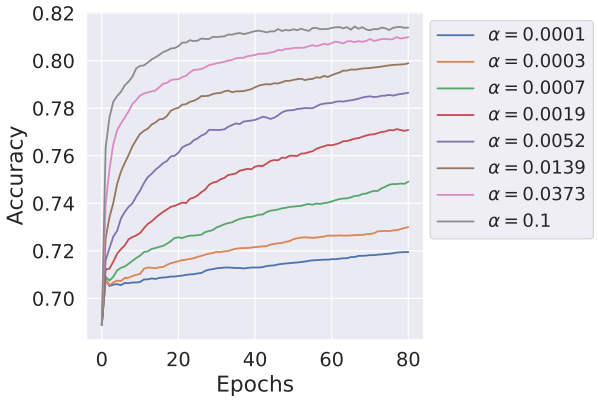


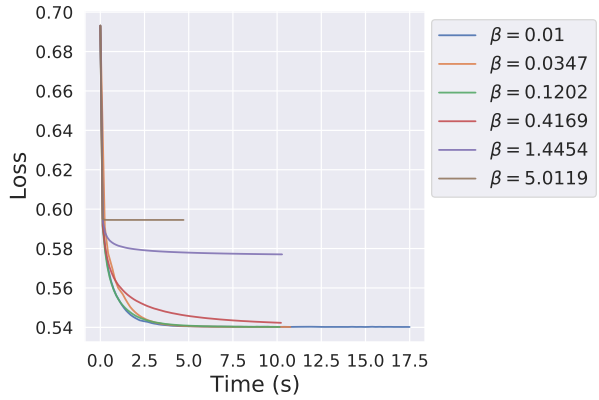


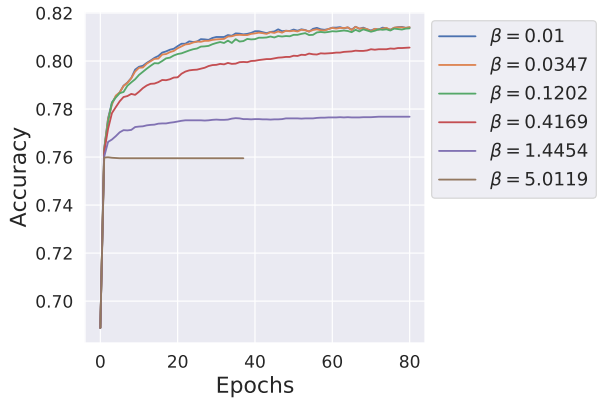


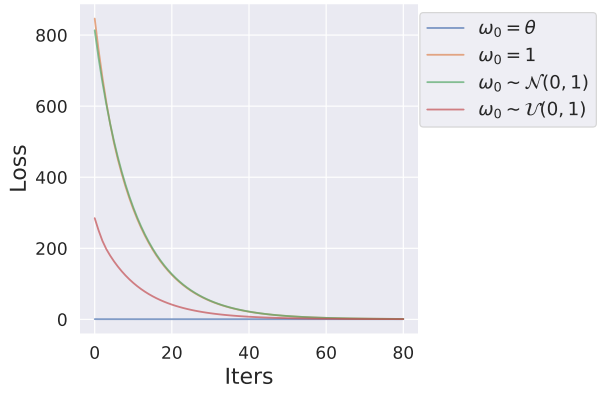
6 Приложение Б. Стохастический градиентный спуск.

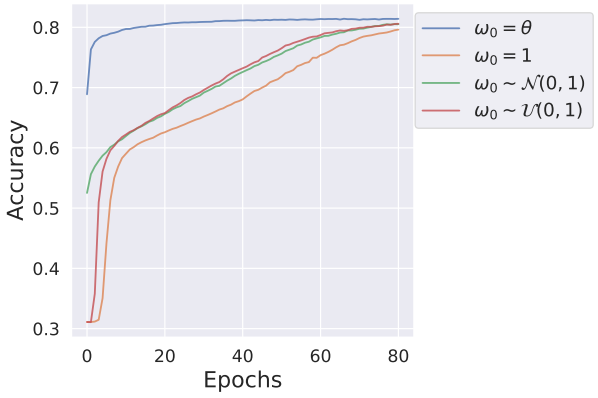


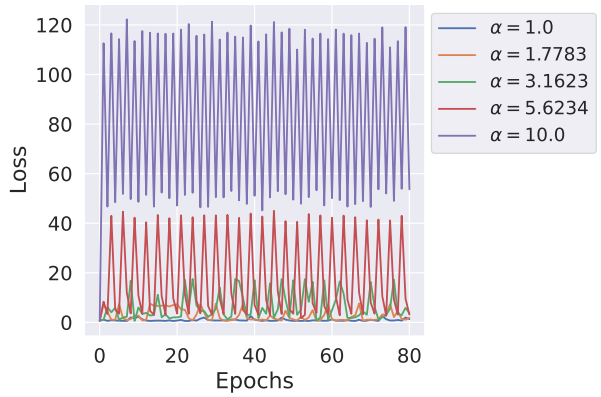


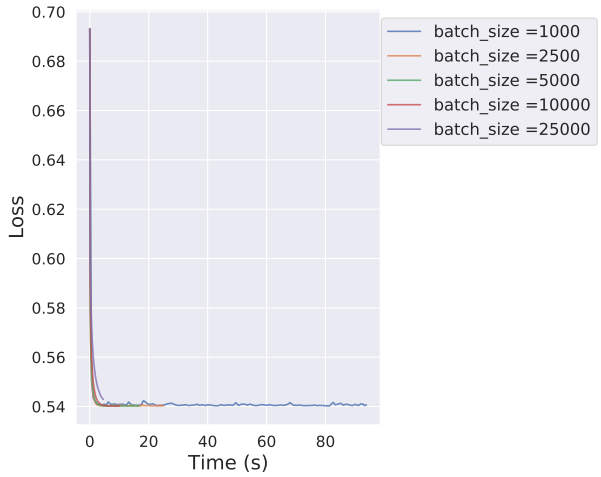

Loss
Time (s)
$\beta = 0.01$
$\beta = 0.0347$
$\beta = 0.1202$
$\beta = 0.4169$
$\beta = 1.4454$
$\beta = 5.0119$

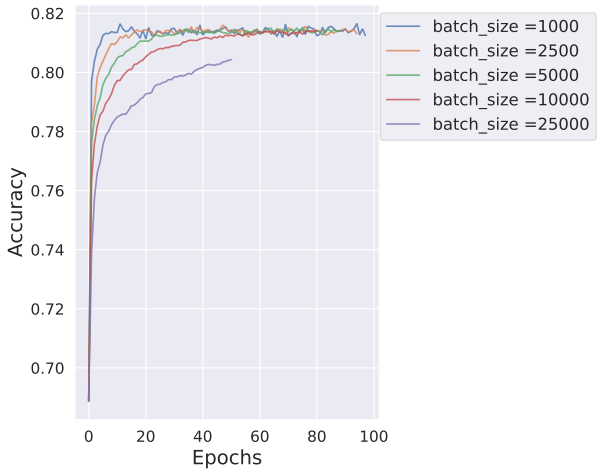

Accuracy
Epochs
$\beta = 0.01$
$\beta = 0.0347$
$\beta = 0.1202$
$\beta = 0.4169$
$\beta = 1.4454$
$\beta = 5.0119$


Loss
Iters
$\omega_0 = \theta$
$\omega_0 = 1$
$\omega_0 \sim \mathcal{N}(0, 1)$
$\omega_0 \sim \mathcal{U}(0, 1)$


Accuracy
Epochs
$\omega_0 = \theta$
$\omega_0 = 1$
$\omega_0 \sim \mathcal{N}(0, 1)$
$\omega_0 \sim \mathcal{U}(0, 1)$


Loss
Epochs
$\alpha = 1.0$
$\alpha = 1.7783$
$\alpha = 3.1623$
$\alpha = 5.6234$
$\alpha = 10.0$


Loss
Time (s)
batch_size =1000
batch_size =2500
batch_size =5000
batch_size =10000
batch_size =25000


Accuracy
Epochs
batch_size =1000
batch_size =2500
batch_size =5000
batch_size =10000
batch_size =25000