Отчет о практическом задании «Градиентные методы обучения линейных моделей. Применение линейных моделей для определения токсичности комментария».

Практикум 317 группы, ММП ВМК МГУ.

Аристеев Олег Алексеевич.

Октябрь 2024.

Содержание

1 Вступление

В данной работе рассматриваются линейные модели классификации и градиентные методы обучения. Эксперименты проводились на датасете, содержащим комментарии из раздела обсуждений английской Википедии, который был преобразован для решения задачи бинарной классификации: является комментарий токсичным или нет. В качестве метрик качества использовались *Accuracy*, равная доле правильных предсказаний среди всех

2 Теоретическая часть

Пусть $(x_i, y_i)_{i=1}^l$ – обучающая выборка, где $x_i \in \mathbb{R}^d$, а $y_i \in Y = \{-1, 1\}$. Тогда линейная модель бинарной классификации будет иметь следующий вид:

$$a(x, w) = \text{sign}\left(\langle w, x\rangle\right)$$

Здесь $w \in \mathbb{R}^d$ – вектор весов. Отступом будет называться следующая величина:

$$M_i(w) = y_i \cdot \langle w, x_i\rangle$$

Если $M_i(w) < 0$, то модель допускает ошибку на объекте x_i, поскольку y_i и $\langle w, x_i\rangle$ имеют разные знаки. Чем больше $M_i(w)$, тем «надёжнее» модель классифицирует объект x_i.

Пусть $\mathscr{L}(M)$ такая монотонно невозрастающая функция, что $[M < 0] \leqslant \mathscr{L}(M)$. Тогда задача обучения линейной модели классификации имеет следующий вид:

$$Q(X, Y, w) = \frac{1}{l}\sum_{i=1}^{l} \mathscr{L}(M_i(w)) \to \min_w$$

2.1 Бинарная логистическая регрессия

В рамках задачи бинарной логистической регрессии рассматривается логарифмическая функция потерь $\mathscr{L}(M) := \log(1 + e^{-M})$. Тогда:

$$Q(X, Y, w) = \frac{1}{l}\sum_{i=1}^{l} \mathscr{L}(M_i(w)) = \frac{1}{l}\sum_{i=1}^{l} \log(1 + \exp(-y_i \langle w, x_i\rangle)))$$

Найдем градиент функции $Q(X, w)$ по переменной w. Поскольку он линеен, задача сводится к поиску градиента функции $\mathscr{L}(M_i(w))$. Для начала найдем её дифференциал:

$$\begin{aligned}
d_w\mathscr{L}(M_i(w))[dw] &= \frac{1}{1 + \exp(-y_i \langle w, x_i\rangle)} d_w[1 + \exp(-y_i \langle w, x_i\rangle)][dw] = \\
&= \frac{\exp(-y_i \langle w, x_i\rangle)}{1 + \exp(-y_i \langle w, x_i\rangle)} d_w[-y_i (\langle w, x_i\rangle)][dw] = \\
&= -y_i \frac{\exp(-y_i \langle w, x_i\rangle)}{1 + \exp(-y_i \langle w, x_i\rangle)} \langle dw, x_i\rangle = \\
&= \left\langle -x_i y_i \frac{\exp(-y_i \langle w, x_i\rangle)}{1 + \exp(-y_i \langle w, x_i\rangle)}, dw \right\rangle
\end{aligned}$$

Значит градиент $\mathscr{L}(M_i(w))$ по вектору w равен:

$$\nabla_w\mathscr{L}(M_i(w)) = -x_i y_i \frac{\exp(-y_i \langle w, x_i\rangle)}{1 + \exp(-y_i \langle w, x_i\rangle)}$$

Возвращаясь к градиенту функции $Q(X, Y, w)$:

$$\nabla_w Q(X, Y, w) = -\frac{1}{l}\sum_{i=1}^{l} x_i y_i \frac{1}{1 + \exp(y_i \langle w, x_i\rangle)} \tag{1}$$

2.2 Мультиномиальная логистическая регрессия

Теперь рассмотрим логистическую регрессию с произвольным числом классов K. Для удобства будем считать $Y = \{1, \dots, K\}$ (ясно, что можно было присвоить названиям классов любые другие метки). Тогда линейный классификатор будет иметь следующий вид:

$$a(x, w) = \arg\max_{i=\overline{1,K}} \langle x, w_i \rangle$$

Здесь $w_1, \dots, w_K \in \mathbb{R}^d$ – векторы весов. Составим из них матрицу $w = \left[w_1 | \dots | w_K\right] \in \mathbb{R}^{d \times K}$. Вероятность, что объект x принадлежит классу $j \in Y$:

$$P(y = j | x, w) = \frac{\exp \langle w_j, x \rangle}{\sum_{k=1}^{K} \exp \langle w_k, x \rangle}$$

По принципу максимального правдоподобия, получим:

$$\text{MSE}(X, Y, w) = \prod_{i=1}^{l} P(y = y_i | x_i, w) \to \max_w$$

Прологарифмируем функцию правдоподобия получим, что задача обучения мультиномиальной логистической регрессии имеет следующий вид:

$$\log \text{MSE}(X, Y, w) = \sum_{i=1}^{l} \log P(y = y_i | x_i, w) \to \max_w$$

Или, в терминах функции потерь, можно записать:

$$Q(X, Y, w) = \frac{1}{l} \sum_{i=1}^{l} \mathscr{L}(x_i, w) = \frac{1}{l} \sum_{i=1}^{l} \Big[- \log P(y = y_i | x_i, w) \Big] \to \min_w$$

Тогда найдем дифференциал общего члена $\mathscr{L}(x_i, w)$ по переменной $w = [w_1, \dots, w_K] \in \mathbb{R}^{d \times K}$ с приращением $dw = [dw_1, \dots, dw_K] \in \mathbb{R}^{d \times K}$:

$$d_w \mathscr{L}(x_i, w)[dw] = d_w \Big[- \log P(y = y_i | x_i, w) \Big][dw] =$$

$$= d_w \left[- \log \frac{\exp \langle w_{y_i}, x_i \rangle}{\sum_{k=1}^{K} \exp \langle w_k, x_i \rangle} \right][dw] = d_w \left[\log \sum_{k=1}^{K} \exp \langle w_k, x_i \rangle - \log \left(\exp \langle w_{y_i}, x_i \rangle \right) \right][dw] =$$

$$= d_w \left[\log \sum_{k=1}^{K} \exp \langle w_k, x_i \rangle - \langle w_{y_i}, x_i \rangle \right][dw] = d_w \left[\log \sum_{k=1}^{K} \exp \langle w_k, x_i \rangle \right][dw] - \langle dw_{y_i}, x_i \rangle =$$

$$= \frac{1}{\sum_{k=1}^{K} \exp \langle w_k, x_i \rangle} d_w \left[\sum_{k=1}^{K} \exp \langle w_k, x_i \rangle \right][dw] - \langle dw_{y_i}, x_i \rangle =$$

$$= \frac{1}{\sum_{j=1}^{K} \exp \langle w_j, x_i \rangle} \sum_{k=1}^{K} d_w \left[\exp \langle w_k, x_i \rangle \right][dw] - \langle dw_{y_i}, x_i \rangle =$$

$$= \sum_{k=1}^{K} \left(\underbrace{\frac{\exp \langle w_k, x_i \rangle}{\sum_{j=1}^{K} \exp \langle w_j, x_i \rangle}}_{P(y=k | x_i, w)} \cdot \langle dw_k, x_i \rangle \right) - \langle dw_{y_i}, x_i \rangle =$$

$$= \sum_{k=1}^{K} P(y = k|x_i, w) \langle dw_k, x_i \rangle - \langle dw_{y_i}, x_i \rangle =$$

$$= \sum_{k=1}^{K} \left\langle \underbrace{\Big(P(y = k|x_i, w) - \mathbb{1}\{k = y_i\}\Big) x_i}_{A_k}, dw_k \right\rangle =$$

Заметим, что последнее выражение можно переписать в форме матричного скалярного произведения, где $A = [A_1, \ldots, A_K]$:

$$= \sum_{k=1}^{K} \langle A_k, dw_k \rangle = \operatorname{tr}\left((dw)^T A\right) = \langle A, dw \rangle$$

Это означает, что матрица A равна градиенту функции $\mathscr{L}(x_i, w)$. Таким образом, k-й столбец градиента функции $\mathscr{L}(x_i, w)$ равен:

$$\big(\nabla_w \mathscr{L}(x_i, w)\big)_k = \left(P(y = k|x_i, w) - \mathbb{1}\{k = y_i\}\right) x_i$$

В силу линейности градиента, получим:

$$\nabla_w Q(X, Y, w) = \frac{1}{l} \sum_{i=1}^{l} \nabla_w \mathscr{L}(x_i, w)$$

Значит, k-й столбец градиента функции $Q(X, Y, w)$:

$$\big(\nabla_w Q(X, Y, w)\big)_k = \frac{1}{l} \sum_{i=1}^{l} \left(\frac{\exp \langle w_k, x_i \rangle}{\sum_{j=1}^{K} \exp \langle w_j, x_i \rangle} - \mathbb{1}\{k = y_i\} \right) x_i \tag{2}$$

2.3 Эквивалентность бинарной логистической регрессии и мультиномиальной с двумя классами

Рассмотрим задачу бинарной логистической регрессии:

$$a_1(x, w) = \operatorname{sign}\left(\langle w, x \rangle\right)$$

И мультиномиальную логистическую регрессию при $Y = \{-1, 1\}$:

$$a_2(x, w) = \arg\max_{y \in Y} \langle x, w_y \rangle$$

Пусть $\hat{w}$ – набор весов для модели a_1. Тогда заметим, что:

$$a_1(x, \hat{w}) = \operatorname{sign}\left(\langle x, \hat{w} \rangle\right) = \arg\max\{\underbrace{0}_{y=-1}, \underbrace{\langle x, \hat{w} \rangle}_{y=1}\} = \arg\max\{\underbrace{\langle x, 0 \rangle}_{y=-1}, \underbrace{\langle x, \hat{w} \rangle}_{y=1}\} =$$

Выберем в качестве $w_{-1} := 0$, а $w_{+1} := \hat{w}$:

$$= \arg\max\{\underbrace{\langle x, w_{-1} \rangle}_{y=-1}, \underbrace{\langle x, w_{+1} \rangle}_{y=1}\} = \arg\max_{y \in Y} \langle x, w_y \rangle = a_2(x, [0, \hat{w}])$$

Теперь в другую сторону. Пусть $[w_{-1}, w_{+1}]$ – совокупность весов для модели a_2. Тогда:

$$a_2(x, [w_{-1}, w_{+1}]) = \arg\max_{y \in Y} \langle x, w_y \rangle = \arg\max\{\langle x, w_{-1} \rangle, \langle x, w_{+1} \rangle\} =$$

Выберем в качестве $w := w_{+1} - w_{-1}$

$$= \operatorname{sign}\left(\langle w_{+1}, x \rangle - \langle w_{-1}, x \rangle\right) = \operatorname{sign}\left(\langle w_{+1} - w_{-1}, x \rangle\right) = a_1(x, w_{+1} - w_{-1})$$

Теперь докажем, что если набор весов w является нулем градиента функции потерь бинарной логистической регрессии Q_1 (1), то совокупность весов $[0, w]$ будет нулем градиента функции потерь мультиномиальной логистической регрессии с двумя классами Q_2 (2) и наоборот. Рассмотрим:

$$\nabla_w Q_1(X, Y, w) = -\frac{1}{l}\sum_{i=1}^{l} x_i y_i \frac{1}{1 + \exp(y_i \langle w, x_i \rangle)}$$

Заметим, что:

$$\mathbb{1}\{y_i = -1\} = \begin{cases} 1, & y_i = -1, \\ 0, & y_i = 1 \end{cases} \Longrightarrow \mathbb{1}\{y_i = -1\} = \frac{1 - y_i}{2}$$

$$\mathbb{1}\{y_i = +1\} = \begin{cases} 1, & y_i = 1, \\ 0, & y_i = -1 \end{cases} \Longrightarrow \mathbb{1}\{y_i = +1\} = \frac{1 + y_i}{2}$$

Тогда:

$$\nabla_w Q_1(X, Y, w) = -\frac{1}{l}\sum_{i=1}^{l}\left(x_i y_i \frac{\exp(-y_i \langle w, x_i \rangle)}{1 + \exp(-y_i \langle w, x_i \rangle)}\mathbb{1}\{y_i = -1\} + x_i y_i \frac{\exp(-y_i \langle w, x_i \rangle)}{1 + \exp(-y_i \langle w, x_i \rangle)}\mathbb{1}\{y_i = +1\}\right) =$$

$$= -\frac{1}{l}\sum_{i=1}^{l}\left(-x_i \frac{\exp(\langle w, x_i \rangle)}{1 + \exp(\langle w, x_i \rangle)}\frac{1 - y_i}{2} + x_i \frac{\exp(-\langle w, x_i \rangle)}{1 + \exp(-\langle w, x_i \rangle)}\frac{1 + y_i}{2}\right) =$$

$$= \frac{1}{2l}\sum_{i=1}^{l}\left(\frac{\exp(\langle w, x_i \rangle)(1 - y_i)}{1 + \exp(\langle w, x_i \rangle)} - \frac{1 + y_i}{1 + \exp(-\langle w, x_i \rangle)}\right) x_i =$$

$$= \frac{1}{2l}\sum_{i=1}^{l}\left(\frac{\exp(\langle w, x_i \rangle)(1 - y_i) - 1 - y_i)}{1 + \exp(\langle w, x_i \rangle)}\right) x_i$$

Теперь рассмотрим градиент функции $Q_2(X, Y, w)$ для задачи многомерной логистической регрессии. Он будет состоять из двух компонент, каждый соответствующий классу $y = 1$ или $y = -1$. Рассмотрим первую компоненту, отвечающую классу -1:

$$\Big[\nabla_w Q_2(X, Y, w)\Big]_{y=-1} = \frac{1}{l}\sum_{i=1}^{l}\left(\frac{\exp\langle w_{-1}, x_i \rangle}{\exp\langle w_{-1}, x_i \rangle + \exp\langle w_{+1}, x_i \rangle} - \mathbb{1}\{y_i = -1\}\right) x_i =$$

Пользуясь тем, что $w_{-1} = 0$, $w_{+1} = w$:

$$= \frac{1}{l}\sum_{i=1}^{l}\left(\frac{1}{1 + \exp\langle w, x_i \rangle} - \mathbb{1}\{y_i = -1\}\right) x_i = \frac{1}{l}\sum_{i=1}^{l}\left(\frac{1}{1 + \exp\langle w, x_i \rangle} + \frac{y_i - 1}{2}\right) x_i$$

$$= \frac{1}{2l}\sum_{i=1}^{l}\left(\frac{2 + (y_i - 1)(1 + \exp\langle w, x_i \rangle)}{1 + \exp\langle w, x_i \rangle}\right) x_i = \frac{1}{2l}\sum_{i=1}^{l}\left(\frac{y_i + 1 + (y_i - 1)\exp\langle w, x_i \rangle)}{1 + \exp\langle w, x_i \rangle}\right) x_i = -\nabla_w Q_1(X, Y, w)$$

Теперь рассмотрим вторую компоненту градиента:

$$\Big[\nabla_w Q_2(X, Y, w)\Big]_{y=+1} = \frac{1}{l}\sum_{i=1}^{l}\left(\frac{\exp\langle w_{+1}, x_i \rangle}{\exp\langle w_{-1}, x_i \rangle + \exp\langle w_{+1}, x_i \rangle} - \mathbb{1}\{y_i = +1\}\right) x_i =$$

$$= \frac{1}{l}\sum_{i=1}^{l}\left(\frac{\exp\langle w, x_i \rangle}{1 + \exp\langle w, x_i \rangle} - \frac{1 + y_i}{2}\right) x_i = -\frac{1}{2l}\sum_{i=1}^{l}\left(\frac{y_i + 1 + (y_i - 1)\exp\langle w, x_i \rangle)}{1 + \exp\langle w, x_i \rangle}\right) x_i = \nabla_w Q_1(X, Y, w)$$

Таким образом, были установлены следующие равенства:

$$\Big[\nabla_w Q_2(X, Y, w)\Big]_{y=-1} = -\nabla_w Q_1(X, Y, w)$$
$$\Big[\nabla_w Q_2(X, Y, w)\Big]_{y=+1} = \nabla_w Q_1(X, Y, w)$$

3 Эксперименты

Исходный набор данных состоит из двух частей: обучающей и тестовой выборки, которые содержат 52061 и 20676 объектов соответственно. Каждый объект – документ – представляет из себя строку, а его метка может принимать одно из двух значений $\{+1, -1\}$, которые соответствуют тому, токсичен комментарий или нет.

Наблюдается дисбаланс в сторону нетоксичных комментариев: их 69.11% от общего числа. Такое соотношение сохраняется как в обучающей (68.83%), так и в тестовой выборке (69.81%). Поэтому константное приближение классом -1 будет иметь **Accuracy = 0.6981**.

3.1 Предварительная обработка текста

Для начала весь текст был обработан. Каждый комментарий был приведен к нижнему регистру, все символы, не являющиеся буквами и цифрами, были заменены на пробелы. Это было сделано, потому что в рамках текущего и части последующих экспериментов каждый документ будет представляться в качестве d-мерного вектора $x = [x_1, \ldots, x_d]$, где x_j – количество встретившихся в данном документе слов с порядковом номером j, а d – длина словаря. Кроме того, в конце будет рассмотрено другое представление с использование `Tfidf`.

3.2 Преобразование выборки в разреженную матрицу

В соответствии с предыдущим пунктом, выборка была преобразована в разреженную матрицу с помощью конструктора `sklearn.feature_extraction.text.CountVectorizer`. Таким образом, матрица объекты-признаки на (i, j)-позиции содержит количество слов с порядковым номером j в i-м документе.

Кроме того, использовался параметр `min_df`, отвечающий за пороговое значение частоты слова для добавления его в словарь (табл. 1)

`min_df`	1	2	5	10	20	50	100	500	1000	5000	10000
Длина словаря d	90764	37837	18227	11222	7180	3928	2337	586	303	60	21

Таблица 1: Зависимость длины словаря d от порога `min_df`

При малых значениях параметра `min_df` признаковое описание документа становится очень длинным, что может привести к вычислительным трудностям и переобучению, при больших – уменьшается, из-за чего может потеряться важная информация, содержащаяся в документе. В дальнейших экспериментах будет рассмотрена зависимость качества решения задачи от этого и других параметров.

3.3 Градиентный спуск и стохастический градиентный спуск. Сравнение численного и аналитического подсчета градиента

В рамках исследования были реализованы методы градиентного спуска (GD) и стохастического градиентного спуска (SGD). Градиент функции потерь $Q(X, Y, w) = \frac{1}{n}\sum_{i=1}^{n} \log(1 + \exp(-y_i \langle w, x_i \rangle))$ можно было вычислять двумя способами: либо по аналитической формуле, полученной в теоретической части (1):

$$\nabla_w Q(X, Y, w) = -\frac{1}{n}\sum_{i=1}^{n} x_i y_i \frac{1}{1 + \exp(y_i \langle w, x_i \rangle)},$$

либо «численно», пользуясь методом конечных разностей:

$$[\nabla_w Q(X, y, w)]_i \approx \frac{Q(X, y, w + \varepsilon e_i) - Q(X, y, w)}{\varepsilon}, \quad i = 1, \ldots, d$$

здесь $x_1, ..., x_n \in \mathbb{R}^d$ – объекты выборки, $y \in \mathbb{R}^n$ – столбец ответов и $w \in \mathbb{R}^d$ – набор весов. Если считать, что арифметические операции, а также операции логарифмирования и экспонирования требуют константного времени, то вычисление скалярного произведения $\langle w, x_i \rangle$ будет требовать $O(d)$ времени, а значит сложность вычисления функции потерь $Q(X, Y, w)$ составит $O(nd)$. Аналитическое вычисление градиента $t_{analytical}$ также имеет сложность $O(nd)$. В то же время «численное» вычисление градиента требует $2d$ раз вычислить значение функции потерь, а значит асимптотическая сложность такого подхода – $t_{numerical} \sim O(2d \cdot nd) = O(nd^2)$, что на порядок d больше, чем аналитическое вычисление $t_{analytical}$. А если учитывать, что значение длины словаря d может быть достаточно большим, например, 90764 при выборе `min_df = 1`, то подсчет численного градиента может требовать значительно большего вычислительного времени, чем аналитическое. Этот теоретический результат подтверждается экспериментально:

d	$\mathbf{t_{analytical}}$ **(мс)**	$\mathbf{t_{numerical}}$ **(мс)**
21	5.245	7.523
60	7.004	26.816
303	10.412	123.713
586	12.060	232.749
2337	14.141	868.585

Таблица 2: Среднее время итерации градиентного спуска при различных значениях длины словаря d

Действительно, по мере роста d время, необходимое для вычисления градиента функции потерь, растет быстрее при численном расчете, чем при аналитическом. Коэффициент корреляции на полученных данных (см. 2) между d и $t_{numerical}/t_{analytical}$ равен 0.998, что согласуется с асимптотическими оценками для сложности вычисления. Рассчеты выполнялись при использовании обычного градиентного спуска со следующими параметрами: `step_alpha = 1, step_beta = 0, tolerance = 1e-5, max_iter = 1000, l2_coef = 0`.

3.4 Зависимость градиентного спуска от параметров

В этом эксперименте был выбран `min_df = 0.025`, то есть в словарь не вошли все слова, которые встретились менее, чем в 2.5% от всех документов, `tolerance = 1e-5, max_iter = 1000, l2_coef = 0.01`. Было исследовано поведение градиентного спуска при разных значениях параметров шага `step_alpha` $= \alpha$, `step_beta` $= \beta$ и начального приближения весов w_0.

В рамках текущей реализации градиентного спуска, learing rate определялся по формуле $\eta_k = \alpha/k^\beta$, где k – номер шага градиентного спуска. Таким образом, чем больше β, тем быстрее уменьшается η_k, и тем менее резким может стать изменение функции потерь на каждом шаге. В свою очередь, слишком маленький α может привести к тому, что спуск будет медленным, а чрезмерно большой – к тому, что оптимальный набор весов не будет достигнут. Это подтверждается экспериментально (рис. 1).

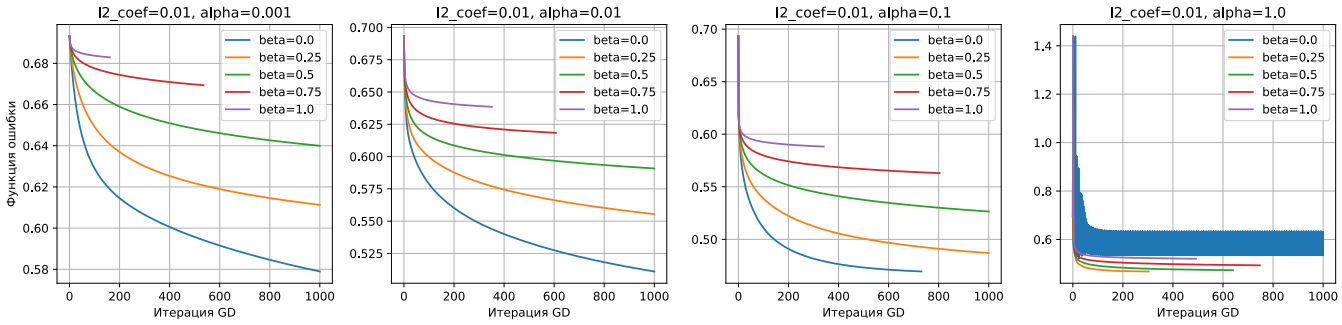


Рис. 1: Зависимость значения функционала ошибок на итерациях GD при разных α и β

Действительно, на первых трёх изображениях видно, что с увеличением β функция ошибок медленнее достигает своего оптимального значения. И исходя из первых трёх экспериментов при $\alpha = 0.001, 0.01, 0.1$ может показаться, что всегда выгоднее выбирать β как можно меньше, однако это не так. При $\alpha = 1$ и $\beta = 0$ наблюдается ситуация, описываемая выше – функция не достигает оптимального значения – её значения колеблются вокруг минимума. Дело в том, что при $\beta = 0$ learning rate не изменяется в зависимости от шага и принимает константное значение, равное α. И если α будет достаточно велико, то функция будет постоянно «перескакивать» вокруг оптимального значения либо вовсе расходиться.

Далее, аналогичный эксперимент, только теперь рассматривается значение *Accuracy* на различных этапах градиентного спуска: (рис 2)

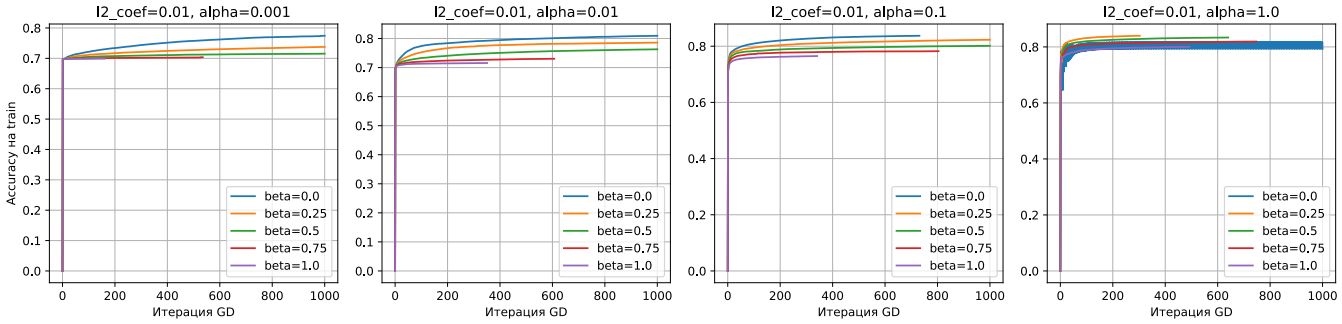


Рис. 2: Зависимость Accuracy на обучающей выборке от номера итерациии GD при разных α и β

Как можно, видеть, при выбранных параметрах градиентного спуска, значение *Accuracy* не претерпевает больших изменений в течение итераций градиентного спуска.

Теперь перейдем к исследованию зависимости поведения градиентного спуска от выбора начальных весов w_0. Будем использовать три способа: $w_0 \in N(0, \sigma), w_0 \in U(-a, a), w_0 = 0$. И исследуем зависимость при фиксированных `step_alpha = 1, step_beta = 0.25, l2_coef = 0.01`. Были подобраны разные значения параметров распределения σ, a (рис. 3):

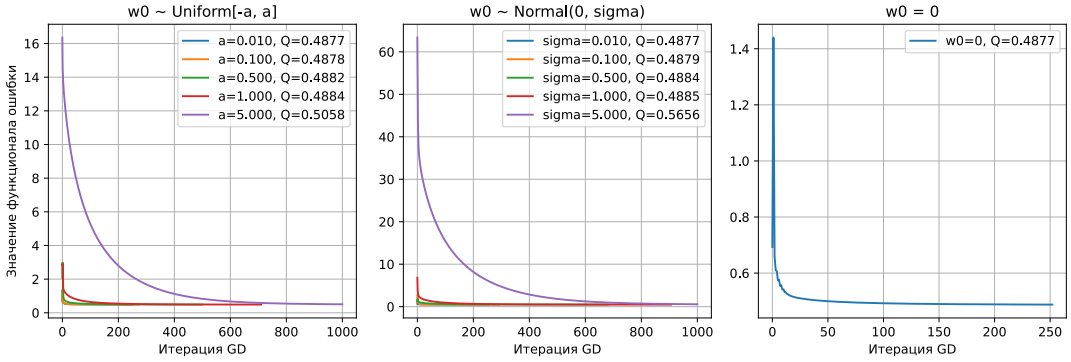


Рис. 3: Зависимость значения функционала ошибок от номера итерациии GD при разных w_0

Как можно видеть, если выбрать параметры распределения достаточно большими, то на начальном наборе весов, функция ошибок принимает достаточно большие значения. И поэтому у неё происходит более «резкий» спуск. Однако на текущих данных было замечено, что чем меньше параметр распределения, тем меньше получается итоговое значение функции, и тем больше распределение похоже на $w_0 = 0$. Наименьшее значение функционала шибки было получено при выбор весов $w_0 = 0$ (рис. 4):

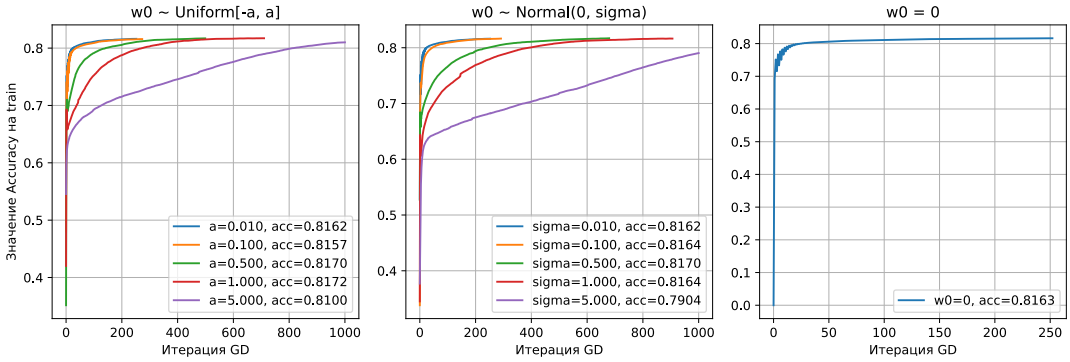


Рис. 4: Зависимость значения accuracy на train от номера итерациии GD при разных w_0

3.5 Зависимость стохастического градиентного спуска от параметров

Теперь, проведем такое же исследование для стохастического градиентного спуска. Зафиксируем параметры стохастического градиентного спуска: `tolerance = 1e-5, max_iter = 100, l2_coef = 0.01`. И исследуем зависимость от `step_alpha` $= \alpha$, `step_beta` $= \beta$. Были получены следующие значения (рис. 5, 6):

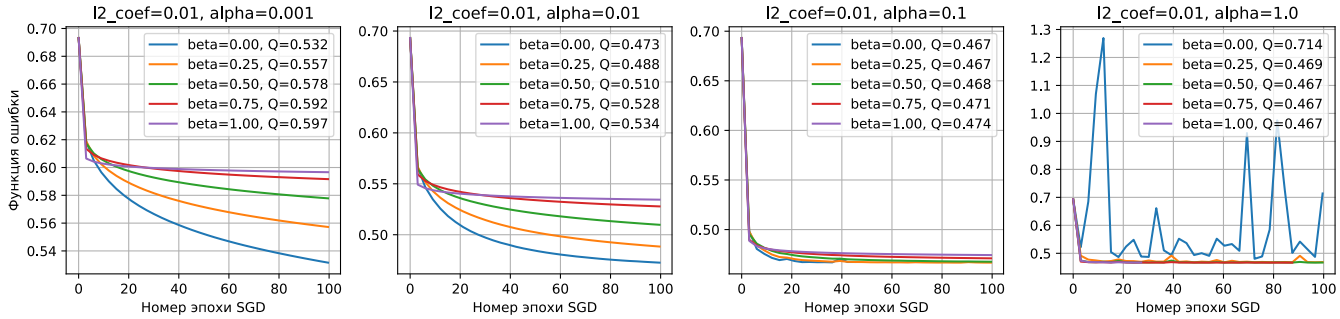


Рис. 5: Зависимость значения функции ошибок от номера эпохи SGD при разных α и β

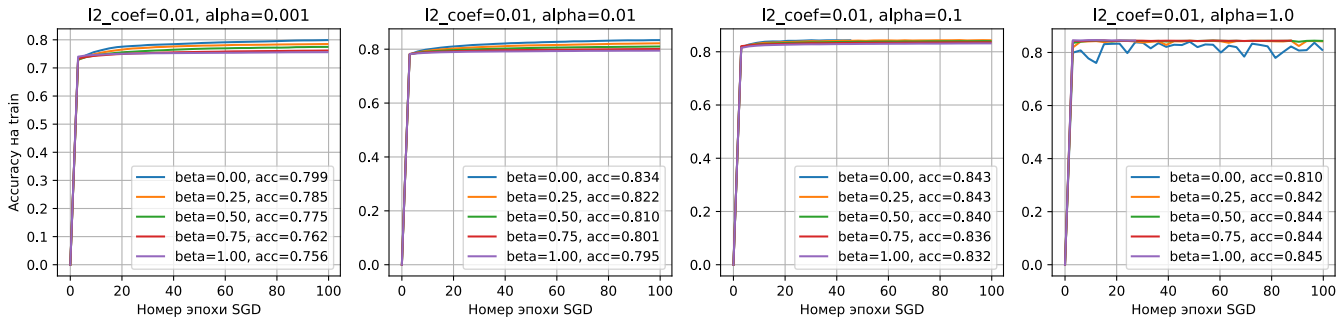


Рис. 6: Зависимость значения accuracy на train от номера эпохи SGD при разных α и β

Выводы, которые были сделаны для параметров α и β обычного градиентного спуска переносятся также на стохастический градиентный спуск. Основное различие заключается в том, что функция менее плавно достигает своего минимума, а чаще делает скачки в процессе. Теперь проведем исследование зависимости от `batch_size`:

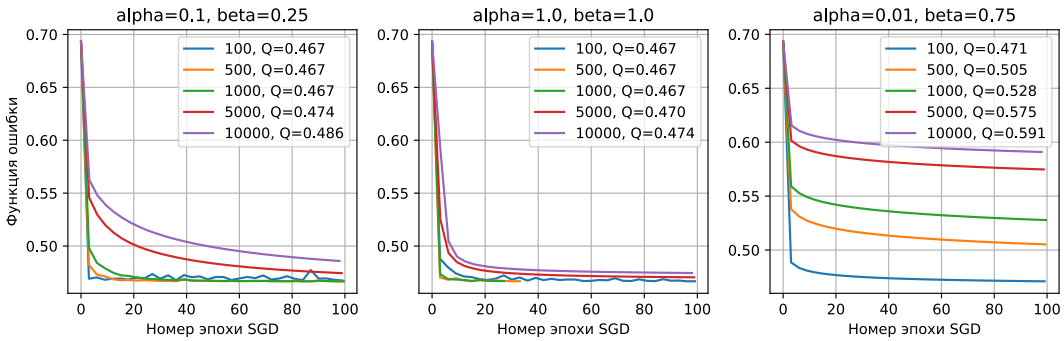


Рис. 7: Зависимость значения функции ошибок от номера эпохи SGD при разных `batch_size`

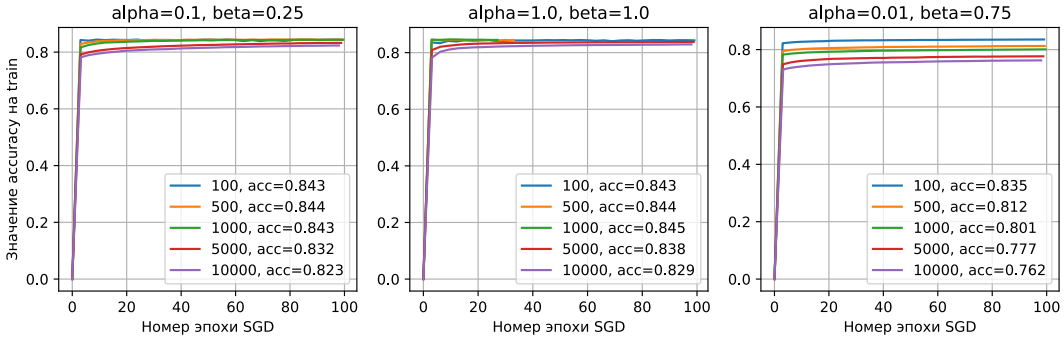


Рис. 8: Зависимость значения accuracy на train от номера эпохи SGD при разных `batch_size`

Видно, что, чем меньше `batch_size`, тем дольше функция сходится к своему оптимальному значению. Однако, также с уменьшением `batch_size` увеличивается количество вычислений градиента, что приводит к вычислительным и временным затратам.

Теперь найдем зависимость от начального приближения весов w_0. Как и в прошлом эксперименте, были рассмотрены три случая: $w_0 = 0, w_0 \in N(0, \sigma), w_0 \in U[-a, a]$. Результаты эксперимента отмечены на (рис. 9, 10):

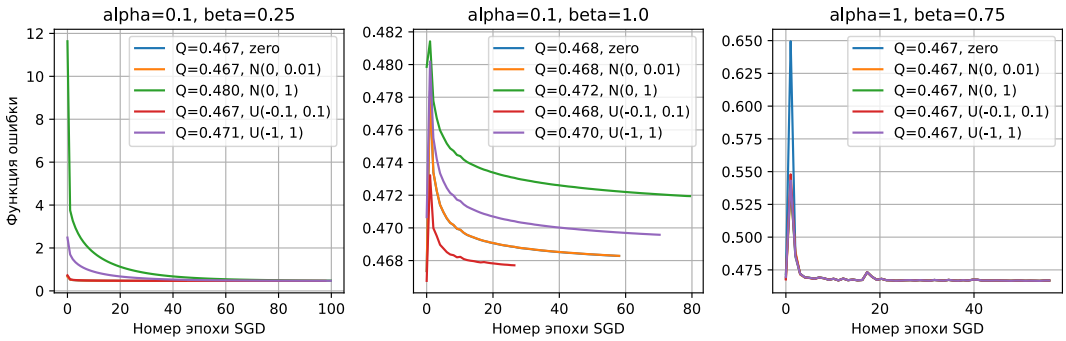


Рис. 9: Зависимость значения функции ошибок от номера эпохи SGD при разных w_0

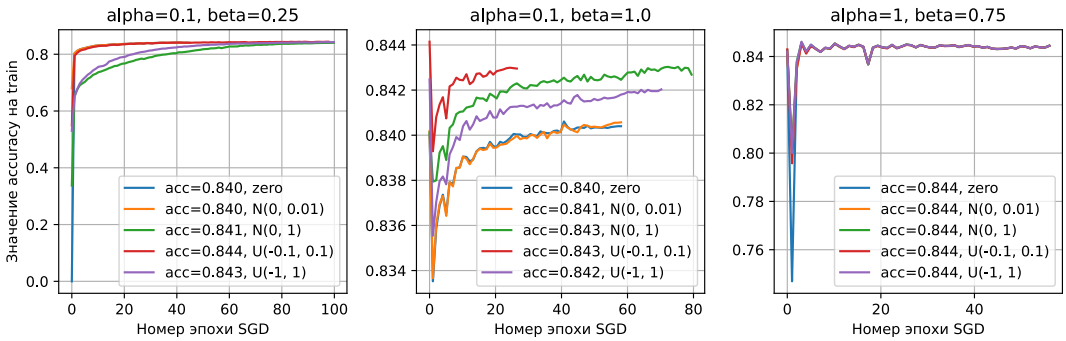


Рис. 10: Зависимость значения accuracy на train от номера эпохи SGD при разном w_0

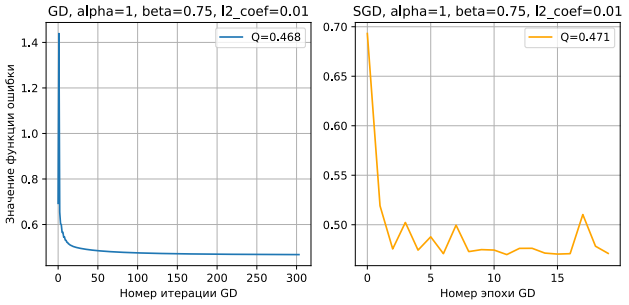
Исходя из графиков, можно сделать вывод, что наилучшая точность была получена при выборе $w_0 \in U[-0.1, 0.1]$.

3.6 Сравнение градиентного и стохастического градиентного спуска

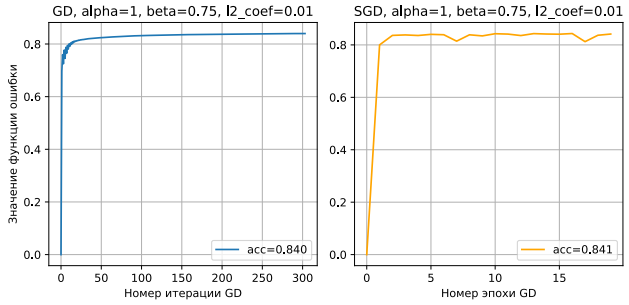
Сравнение этих методов проведем по среднему времени работы и итоговому качеству. `max_iter_GD` – максимальное количество итераций в градиентном спуске, `max_iter_SGD` – максимальное количество итераций в стохастическом градиентном спуске. Тогда вычисление градиента в первом методе происходит `max_iter_GD` раз, а во втором – около `max_iter_SGD` $\times$ `N/batch_size` раз, где `N` – количество объектов в обучающей выборке. Поэтому сравнение по времени можно проводить, при

$$\texttt{max\_iter\_GD} \approx \texttt{max\_iter\_SGD} \times \texttt{N/batch\_size}$$

По условию, `N = 52061`, выберем `max_iter_GD=1000, batch_size=1000`, тогда, чтобы количество вычислений градиента оказалось одинаковым, можно выбрать `max_iter_SGD = 20`. При `step_alpha=1, step_beta=0.25, l2_coef=0.01` время выполнение GD заняло **3.580 (c)**, а SGD – **0.0329 (c)** (рис. 11):



(a) Зависимость значения функции ошибок от номера итерации GD и эпохи SGD



(b) Зависимость значения accuracy на train от номера итерации GD и эпохи SGD

Рис. 11: Сравнение GD и SGD

Как можно видеть, алгоритм стохастического спуска работает в **108.6** раз быстрее при выбранных параметрах. Однако стохастический спуск немного уступает обычному в Accuracy на test – **0.7899** и **0.79618** соответственно.

3.7 Алгоритм лемматизации, удаление стоп-слов

В рамках этого и последующих экспериментов обучающая выборка была разбита на валидационную и обучающую в отношении 3:7, чтобы избежать утечки данных.

В этом эксперименте было рассмотрено влияние применение алгоритма лемматизации и удаления стоп-слов на качество (Accuracy), время обучения и размер признакового описания при решении задачи. Лемматизация — это процесс приведения слова к его базовой форме, которая называется леммой. Лемма обычно соответствует начальной форме слова например: «write», «written», «wrote» преобразуются в форму «write».

Из текста были удалены все стоп-слова. Это часто встречающиеся слова, которые обычно не несут смысловой нагрузки в тексте. Они включают предлоги, союзы, местоимения, артикли и другие части речи, которые часто используются для связки слов и предложений, но сами по себе не добавляют значимой информации, например: the, a, is, in, and, of, to и другие. Кроме того, из текста были убраны все цифры.

Таким образом, количество уникальных слов в тексте сократилось с 90764 до 74755. Исследуем, как предобработка документа повлияет на точность классификации и время работы алгоритма. Для этого выберем одинаковые параметры стохастического градиентного спуска и рассмотрим, как меняется точность классификации в зависимости от того, используется предобработка, или нет. Обучение происходило с стохастического градиентного спуска при `batch_size = 1000, min_df = 1`. Обозначим $\mathbf{time_1}$, $\mathbf{time_2}$ – время обучения на непредобработанной выборке, и на выборке с применением лемматизации и удалением стоп слов соответственно. Аналогично – $\mathbf{Accuracy_1}$, $\mathbf{Accuracy_2}$ – точности модели на **валидационной** выборке, полученные на непредобработанной и предобработанной соответственно обучающей выборке (табл. 3).

step_alpha	step_beta	l2_coef	$\mathbf{time_1}$ (c)	$\mathbf{time_2}$ (c)	$\mathbf{Accuracy_1}$	$\mathbf{Accuracy_2}$
0.10	0.01	0.01	0.039635	**0.037847**	0.860106	**0.872399**
0.05	0.02	0.02	0.054764	**0.048856**	0.849926	**0.862859**
0.25	0.25	0.10	0.014099	**0.006971**	0.827966	**0.836481**
1.00	0.00	0.10	0.060769	**0.043400**	0.618413	**0.829246**

Таблица 3: Результаты перебора параметров для моделей с лемматизацией и удалением стоп-слов и без

Таким образом, на всех проведенных экспериментах модель с вышеуказанной предобработкой данных показывает лучшее качество классификации валидационной выборке. и скорость обучения на

3.8 Анализ влияние других параметров – формата представления и порогов частоты для добавления в словарь.

эксперимент направлен на изучение зависимости от параметров конструктора `CountVectorizer` – `min_df`, `max_df`, и формата представления данных – через `BagOfWords` или `Tfidf`. В прошлом эксперименте было установлено, что использование предобработки тестовых данных улучшает качество на валидацинной выборке, таким образом из выборки были удалены стоп-слова, а значит большая часть слишком частых слов. Поэтому ограничимся исследование только параметра `min_df`.

min_df	$\mathbf{Accuracy_{BoW}}$	$\mathbf{Accuracy_{tfidf}}$	$\mathbf{time_{BoW}}$ (c)	$\mathbf{time_{tfidf}}$ (c)
0.00001	0.863692	**0.898905**	**0.045594**	0.055363
0.00002	0.863628	***0.899033***	0.025229	**0.019038**
0.00005	0.863372	**0.898713**	0.033679	**0.029571**
0.00010	0.862091	**0.896984**	**0.019577**	0.027835
0.00100	0.855497	**0.886228**	**0.015085**	0.023822
0.01000	0.789167	**0.804085**	0.013309	**0.013020**

Таблица 4: Результаты перебора параметров для моделей с использование `BagOfWords` и `Tfidf`

Как видно в таблице 4, использование `Tfidf`-представления улучшает точность на валидационной выборке. Кроме того, максимум точности достигается при `min_df=2e-5`, а при дальнейшем увеличении происходит только убывание.

3.9 Выбор лучшего алгоритма для тестовой выборки

В экспериментах 7-8 было установлено, что наиболее оптимальным является использование лемматизации, удаления стоп-слов и параметра `min_df=2e-5` и представления в формате `Tfidf`. В экспериментах 4-5 было исследовано поведение функционала ошибки и Accuracy на обучающей выборке в зависимости от итерации на градиентом и стохастическом градиентном спуске. Далее были найдены оптимальные значения для `alpha_step, beta_step, l2_score` полным перебором на валидационной выборке по сетке значений.

Итоговыми значениями с наилучшим качеством на валидационной выборке (Accuracy = 0.8998) получились `alpha_step = 0.06, beta_step = 1e-4, l2_score=0.01`. На тестовой выборке при таких параметрах получилось **Accuracy = 0.8615786**, если инициализировать веса $w_0 = 0$. А если действовать в соответствии с результатом четвертого эксперимента и инициализировать $w_0 \in U[-0.1, 0.1]$, то достигается **Accuracy = 0.861724**

Проведем анализ ошибок. Для этого рассмотрим случайную выборку из ложноположительных ответов (FP = 1925) и ложноотрицательных ответов (FN = 588):

Оригинальный текст (нетоксичный)	**После обработки**

`== hi ==`
 	
`screw off!!!!!!!!!!!!!!!!!!!!`
	`['screw']`

`lol this s kinda gay ..... use` `↪ ENCARTA !`	`['kinda', 'gay', 'encarta']`

`"`
 	
`==Vandal killing==`
`That was fast! Thanks!! "`
	`['vandal', 'kill', 'fast']`

`::You people are just sick. |`	`['sick']`
`Wtf, all western corners are 90 degrees.`	`['wtf', 'western', 'corner', 'degree']`

Таблица 5: Примеры ложноположительных распознаваний

Модель могла рассматривать слово «screw» как токсичное, хотя в контексте этого предложения оно может быть просто выражением раздражения, а не прямым оскорблением. Слово «gay» тоже может быть расценено как токсичное. Фраза «Vandal killing» может быть расценена как токсичная, так как она содержит слова «vandal» и «kill», однако в контексте этого предложения она может относиться к удалению вандальных правок, что не является токсичным. Аналогично «Wtf» и «sick».

Оригинальный текст (токсичный)	После обработки
`:::::Nothing makes me happier than watching nazis die.`	`['make', 'happy', 'watch', 'nazis']`
`How many failed marriages/de facto relationships has this bloke had? What a goose.`	`['fail', 'marriage', 'facto', 'relationship', 'bloke', 'goose']`
`islam is nothing but terrorism`	`['islam', 'terrorism']`
`I'm rick james, B1tch!`	`['rick', 'james', 'b', 'tch']`
`I totally agree, religious people are hateful people. If only religion never existed, we would be much better off.`	`['totally', 'agree', 'religious', 'hateful', 'religion', 'exist']`

Таблица 6: Примеры ложнопоотрицательных распознаваний

Первый комментарий содержит прямое желания смерти, но слово «die» является стоп-словом в корпусе `nltk`, поэтому не участвовало при голосовании, также в комментарии присутствует несколько нейтральных слов и положительное слово «happy». Аналогично со вторым и пятым. В четвертом примере буква i была завуалирована символом «!», поэтому оскорбление «B!tch» было распознано как два слова: «B» и «tch».

4 Улучшения

4.1 Добавление n-грамм

В этом эксперименте рассмотрено влияние добавления n-грамм в признаковое описание. Это может быть полезно, потому что многие слова употребляются вместе с другими и вместе могут иметь негативный или позитивный окрас, но по отдельности являются нейтральными или вовсе стоп-словами, например «not good». Тогда оставим все параметры, зафиксированные в 9 эксперименте и добавим 2-граммы, 3-граммы, 4-граммы и 5-граммы зафиксируем точности классификации и длины признакового описания:

N-граммы	Длина признакового описания	Accuracy
1-1	29 748	0.861724
1-2	143 660	0.863271
1-3	184 757	0.863368
1-4	214 520	**0.864142**
1-5	242 330	0.863707

Таблица 7: Влияние n-грамм на длину признакового описания и точность модели

Таким образом, наилучшая точность на тесте достигается при использовании 4-грамм. Получается

Accuracy = 0.864142, однако значительно увеличивается длина признакового описания объекта.

5 Выводы

В ходе выполнения данного практического задания были исследованы градиентные методы обучения линейных моделей, а именно градиентный спуск (GD) и стохастический градиентный спуск (SGD), на примере задачи бинарной классификации токсичности комментариев. Основные выводы по результатам экспериментов можно сформулировать следующим образом:

1. Аналитический расчет градиента значительно эффективнее численного, особенно при больших значениях длины словаря d.

2. Параметры GD и SGD существенно влияют на сходимость и качество модели. Оптимальные значения можно найти экспериментально.

3. Сравнение GD и SGD показало, что SGD значительно быстрее, но требует более тщательного подбора параметров для достижения оптимальной точности.

4. Предобработка данных (лемматизация, удаление стоп-слов) и использование Tfidf улучшают качество модели и сокращают время обучения.