Отчет о практическом задании «Градиентные методы обучения линейных моделей»

Практикум 317 группы, ММП ВМК МГУ

Борисов Иван Максимович

17.11.2023

Содержание

Аннотация

Данное практическое задание посвящено изучению методов градиентного спуска и стохастического градиентного спуска путем рассмотрения зависимости скорости сходимости и точности классификации от гиперпараметров в задаче бинарной классификации эмоциональной окраски текстов.

1 Введение

Любая задача машинного обучения с учителем в конечном итоге сводится к решению *оптимизационной задачи*. Проще говоря, мы хотим, чтобы модель как можно меньше «ошибалась». Для того чтобы измерить, насколько сильно «ошибается» модель, вводится понятия *функционала ошибки*. Тогда любую задачу машинного обучения можно переформулировать следующим образом: необходимо найти такую модель из множества всех допустимых моделей, чтобы заданный функционал ошибки достигал минимального значения — это и есть задача оптимизации.

Вообще говоря, поставленная задача не обязана иметь решение(к примеру, минимум может быть недостижим), но даже если решение и существует, то найти его аналитически удается далеко не всегда.

На помощь приходят численные методы. Основным и на данный момент одним из самых успешных результатов, позволяющих искать хотя бы локальный минимум численно, является *градиентный спуск* (применимый при условии достаточной гладкости исследуемого функционала).

Сам градиентный спуск можно описать как «спуск» шагами некоторой длинны по поверхности, задаваемой введенным функционалом, в сторону наибольшего убывания функции. Таким образом, при правильных длинах шага метод градиентного спуска позволяет с заданной наперед точностью найти один из локальных минимумов.

2 Пояснения к задаче

2.1 Логистическая бинарная функция потерь

Для оценки вероятностей принадлежности объекта к определенному классу можно использовать *логистическую функцию потерь*. Приведем ее вывод и вычислим градиент по вектору весов линейной модели для дальнейшего использования его в методе градиентного спуска.

Пусть задано множество объектов $X = \{x\}_{i=1}^n$ и ответов на них $Y = \{y\}_{i=1}^n$,

$$x_i \in \mathbb{R}^d,\ y_i \in \{0, 1\}$$

Предположим, что $y_i \sim Be(\theta)$, т.е. y_i принадлежат распределению Бернулли с параметром θ.

Запишем функцию правдоподобия для распределения Бернулли:

$$p(y) = \prod_{i=1}^{n} p_i^{y_i}(1 - p_i)^{1-y_i},\ \ p_i = P(y_i = 1)$$

Будем обучать линейную модель $a(x) = \langle w, x\rangle + b$, где $w \in \mathbb{R}^d$ и $b \in \mathbb{R}$, таким образом, чтобы максимизировать функцию правдоподобия.

Предварительно для упрощения записей введем обозначения:

$$w = (w_0 := b, w_1, \ldots, w_d) \quad x_i = (1, x_i^1, \ldots, x_i^d)$$

Таким образом, в новых обозначениях $a(x) = \langle w, x\rangle$.

Теперь заметим, что $\forall x \in \mathbb{R}^{d+1} \Rightarrow a(x) \in \mathbb{R}$, но для максимизации правдоподобия нам необходимо иметь вероятность принадлежности x к положительному классу. Иными словами нужно построить отображение $f : \mathbb{R} \to [0, 1]$. Несложно убедиться, что таким отображением может выступать функция $\sigma(x) = \frac{1}{1+e^{-x}}$.

Теперь все готово к тому, чтобы сформулировать задачу, которая позволит максимизировать функцию правдоподобия при помощи модели $a(x)$:
Пусть $p_i = \sigma(\langle w, x_i \rangle)$ тогда необходимо найти $w = (w_0, w_1, \ldots, w_d)$:

$$p(y|X, w) = \prod_{i=1}^{n} p_i^{y_i}(1 - p_i)^{1-y_i} \to max_w$$

Перейдем от произведения к сумме и от максимизации к минимизации:

$$-\sum_{i=1}^{n} ln\big(p_i^{y_i}(1 - p_i)^{1-y_i}\big) \to min_w$$

$$-\sum_{i=1}^{n} y_i ln p_i + (1 - y_i) ln(1 - p_i) \to min_w$$

Заметим, что $1 - \sigma(x) = \sigma(-x)$, и подставим $p_i = \sigma(\langle w, x_i \rangle)$:

$$-\sum_{i=1}^{n} y_i ln\sigma(\langle w, x_i \rangle) + (1 - y_i) ln(\sigma(-\langle w, x_i \rangle)) \to min_w$$

Пусть теперь $y_i \in \{-1, 1\}$:

$$p(y|X, w) = \prod_{i=1}^{n} p_i^{\frac{y_i+1}{2}}(1 - p_i)^{\frac{1-y_i}{2}} \to max_w$$

$$-\frac{1}{2}\sum_{i=1}^{n}(y_i + 1) ln\sigma(\langle w, x_i \rangle) + (1 - y_i) ln(\sigma(-\langle w, x_i \rangle)) \to min_w$$

Покажем, что $\frac{1}{2}\big((y_i + 1) ln\sigma(\langle w, x_i \rangle) + (1 - y_i) ln(\sigma(-\langle w, x_i \rangle)\big) \Leftrightarrow ln\sigma(y_i \langle w, x_i \rangle)$:

Если $y_i = 1 \Rightarrow \frac{1}{2}\big((y_i + 1) ln\sigma(\langle w, x_i \rangle) + (1 - y_i) ln(\sigma(-\langle w, x_i \rangle)\big) = \frac{1}{\cancel{2}} \cdot \cancel{2} ln\sigma(1 \cdot \langle w, x_i \rangle) = ln\sigma(y_i \langle w, x_i \rangle)$

Если $y_i = -1 \Rightarrow \frac{1}{2}\big((y_i + 1) ln\sigma(\langle w, x_i \rangle) + (1 - y_i) ln(\sigma(-\langle w, x_i \rangle)\big) = \frac{1}{\cancel{2}} \cdot \cancel{2} ln\sigma(-1 \cdot \langle w, x_i \rangle) = ln\sigma(y_i \langle w, x_i \rangle)$

Таким образом:

$$\boxed{Q(w, X, y) = -\sum_{i=1}^{n} ln\sigma(y_i \langle w, x_i \rangle) = \sum_{i=1}^{n} ln(1 + e^{-y_i \langle w, x_i \rangle})}$$

2.1.1 Градиент бинарной логистической функции потерь

Для градиентного спуска нам потребуется знать $\nabla_w Q(w, X, y)$. Найдем его:

$$d\left(\sum_{i=1}^{n} ln(1 + e^{-y_i \langle w, x_i \rangle})\right) = \sum_{i=1}^{n} d\big(ln(1 + e^{-y_i \langle w, x_i \rangle})\big) = \sum_{i=1}^{n} \sigma(-y_i \langle w, x_i \rangle) d\big(e^{-y_i \langle w, x_i \rangle}\big) =$$
$$= \langle \sum_{i=1}^{n} \sigma(-y_i \langle w, x_i \rangle) e^{-y_i \langle w, x_i \rangle} \cdot (-y_i x_i), dw \rangle$$

$$\boxed{\nabla Q = -\sum_{i=1}^{n} \frac{x_i y_i}{1 + e^{-y_i \langle w, x_i \rangle}} e^{-y_i \langle w, x_i \rangle}}$$

2.2 Логистическая многоклассовая функция потерь

Пусть теперь $y_i \in \{0, 1, ..., K\}$. Обобщим предложенный выше подход на случай многоклассовой классификации. Пусть построено K линейных классификаторов $a_k(x) = \langle w_k, x \rangle$, которые оценивают вероятность принадлежности объекта к k-ому классу.

Вместо $\sigma(x) = \frac{1}{1+e^{-x}}$, где $x \in \mathbb{R}$, необходимо найти отображение $f : \mathbb{R}^K \to [0,1]^K$, при условии, что сумма по вектору предсказанных вероятностей равна 1.

В качестве такой функции может выступать $SoftMax(x_0, x_1, \ldots, x_K) = \left(\frac{exp(x_1)}{\sum_{i=0}^{K} exp(x_i)}, \ldots, \frac{exp(x_K)}{\sum_{i=1}^{K} exp(x_i)} \right)$

Тогда $P(y = k|x, w) = \frac{exp(\langle w_k, x \rangle)}{\sum_{i=0}^{K} exp(\langle w_i, x \rangle)}$

Метод максимума правдоподобия:

$$\sum_{i=1}^{n} lnP(y = y_i|x, w) \to max_w$$

Переход к минимуму:

$$\sum_{i=1}^{n} lnP(y = y_i|x, w)^{-1} \to min_w$$

Итого:

$$\boxed{Q(W, X, y) = \sum_{i=1}^{n} ln \frac{\sum_{k=0}^{K} exp(\langle w_i, x_i \rangle)}{exp(\langle w_{y_i}, x_i \rangle)}}$$

2.2.1 Градиент многоклассовой логистической функции потерь

Найдем $\nabla_w Q(W, X, y)$.:

$$d_j \left(\sum_{i=1}^{n} ln \frac{\sum_{k=0}^{K} exp(\langle w_k, x_i \rangle)}{exp(\langle w_{y_i}, x_i \rangle)} \right) = \sum_{i=1}^{n} d_j \left(ln \sum_{k=0}^{K} exp(\langle w_k, x_i \rangle) - \langle w_{y_i}, x_i \rangle \right) =$$

$$= \sum_{i=1}^{n} \left(\frac{exp(\langle w_j, x_i \rangle)}{\sum_{k=0}^{K} exp(\langle w_k, x_i \rangle)} \langle x_i, dw_j \rangle - \langle x_i, dw_{y_i} \rangle [y_i = j] \right) =$$

$$= \sum_{i=1}^{n} \langle (SoftMax_j - [y_i = j]) x_i, dw_j \rangle$$

$$\boxed{\nabla_j Q = \sum_{i=1}^{n} (SoftMax_j - [y_i = j]) x_i, \;\; 0 \le j \le K}$$

2.3 Связь между многоклассовой и бинарной логистическими функциями потерь

Положим в $Q(W, X, y)$ $K = 1$:

$$Q(W, X, y) = \sum_{i=1}^{n} ln \frac{exp(\langle w_0, x_i \rangle) + exp(\langle w_1, x_i \rangle)}{exp(\langle w_{y_i}, x_i \rangle)} =$$
$$= \sum_{i=1}^{n} (y_i ln(1 + exp(\langle w_1 - w_0, x_i \rangle)) + (1 - y_i) ln(1 + exp(\langle w_0 - w_1, x_i \rangle))) =$$
$$= -\sum_{i=1}^{n} \left(y_i ln \left(\frac{1}{1 + exp(\langle -\hat{w}, x_i \rangle)} \right) + (1 - y_i) ln \left(\frac{1}{1 + exp(\langle \hat{w}, x_i \rangle)} \right) \right) =$$
$$= -\sum_{i=1}^{n} y_i ln \sigma(\langle \hat{w}, x_i \rangle) + (1 - y_i) ln \sigma(-\langle \hat{w}, x_i \rangle),$$

где $\hat{w} = w_0 - w_1$

Таким образом, было показано, что многоклассовая логистическая функция потерь обобщает понятие бинарной логистической функции потерь.

2.4 Регуляризация

В данной работе дополнительно к функции потерь добавлена регуляризация:

$$R(w) = \frac{\lambda}{2} \sum_{j=1}^{d} w_j^2, \quad \lambda \in \mathbb{R}$$

Важно отметить, что w_0 не требует регуляризации, поэтому сумма ведется от $j = 1$.

$$\nabla R(w) = (0, \lambda w_1, \ldots, \lambda w_d)$$

Таким образом, градиент функции потерь будет иметь дополнительное слагаемое равное $\nabla R(w)$.

2.5 Градиентный спуск

Пусть задано множество объектов $X = \{x\}_{i=1}^{n}$ и ответов на них $Y = \{y\}_{i=1}^{n}$,

$$x_i \in \mathbb{R}^d, \; y_i \in \{-1, 1\}$$

Будем решать задачу с помощью:

$$a(x) = \langle w, x \rangle \qquad Q(w, X, y) = \frac{1}{n} \sum_{i=1}^{n} ln(1 + e^{-y_i \langle w, x_i \rangle}) + R(w) \to min_w,$$

$$w = (w_0 := b, w_1, \ldots, w_d) \quad x_i = (1, x_i^1, \ldots, x_i^d)$$

Тогда под градиентным спуском понимается нахождение минимума $L(w, X, y)$ по w с наперед заданной точностью с помощью итеративного алгоритма:

$$w^{k+1} = w^k - \eta \nabla_w Q(w, X, Y),$$

$\eta \in \mathbb{R}$ - длина шага

2.5.1 Длина шага (learning rate)

Возникает вопрос — как подбирать длину шага. Существует несколько различных вариантов:

- $exp : \eta_k = \eta_0 \gamma^k$
- $lin : \eta_k = \eta_{max} - \frac{(\eta_{max} - \eta_{min})(k-1)}{n}$, n - макс. число итераций
- $cos : \eta_k = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})(1 + cos(\frac{\pi k}{n}))$
- $inv\ scale : \eta_k = \frac{\alpha}{k^\beta}, \alpha, \beta > 0$

В данной работе использовался вариант *inv scale*, где $\alpha, \beta \in \mathbb{R}_+$ — гиперпраметры модели.

2.5.2 Начальное приближение

Для запуска итеративного алгоритма требуется взять какое-то начальное w^0. В работе будут рассмотрены следующие варианты:

- $w = (0, \ldots, 0)$
- $w \sim N(0, 1)$
- $w \sim U(-1, 1)$
- $w = \left(\frac{\langle f_j, y \rangle}{\langle f_j, f_j \rangle}\right)_{j=1}^{d}, w_0 = 0$, где y - столбец меток, f_j - столбец j-ого признака.
- Обучение по 100 случайным объектам с нулевой начальной инициализацией.

2.5.3 Стохастический градиентный спуск

Стоит уделить внимание сложности алгоритма градиентного спуска.
Пусть t — максимальное число итераций.
Тогда вычислительная сложность градиентного спуска - $O(ndt)$, сложность по памяти - $O(nd)$.

Можно ли как-то улучшить данные параметры?
Вместо обучения на всей выборке, будем обучаться на случайной подвыборке фиксированного размера, такую подвыборку называют *batch (батч)*. Такой метод называют *стохастическим градиентным спуском*, а размер батча становится дополнительным гиперпараметром модели.

Более формально:

$\nabla_w L(w, X, Y) \approx -\sum_{i=1}^{b} \frac{x_i y_i}{1 + e^{-y_i \langle w, x_i \rangle}} e^{-y_i \langle w, x_i \rangle}$, где b — размер батча.

Пусть τ — максимальное число *эпох* (проходов по всей выборке).
Тогда вычислительная сложность градиентного спуска - $O(nd\tau)$, сложность по памяти - $O(bd)$.

На первый взгляд может показаться, что в вычислительной сложности выйгрыша нет, но за проход по одной эпохе веса успевают обновиться $\frac{n}{b}$ раз, что существенно ускоряет скорость сходимости алгоритма.
В памяти теперь нет необходимости хранить всю выборку, достаточно помнить последний батч, а последующие батчи можно динамически подгружать с некоторого носителя.

3 Эксперименты

3.1 Проверка работы градиентного спуска

Сгенерируем случайным образом 1 тыс. объектов с 2 признаками, 1 тыс. ответов и случайным образом инициализируем начальные веса. Все параметры оставим по умолчанию, для стохастического градиентного спуска возьмем размер батча, равный 64.

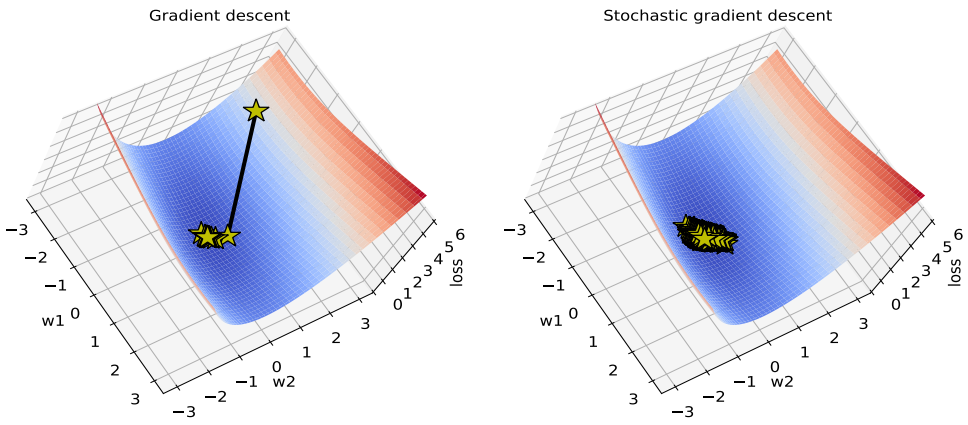

Рис. 1: Градиентные методы

Попробуем уменьшить длину шага, установив $\alpha = 0.1, \beta = 0.5$. Остальные параметры оставим без изменений.

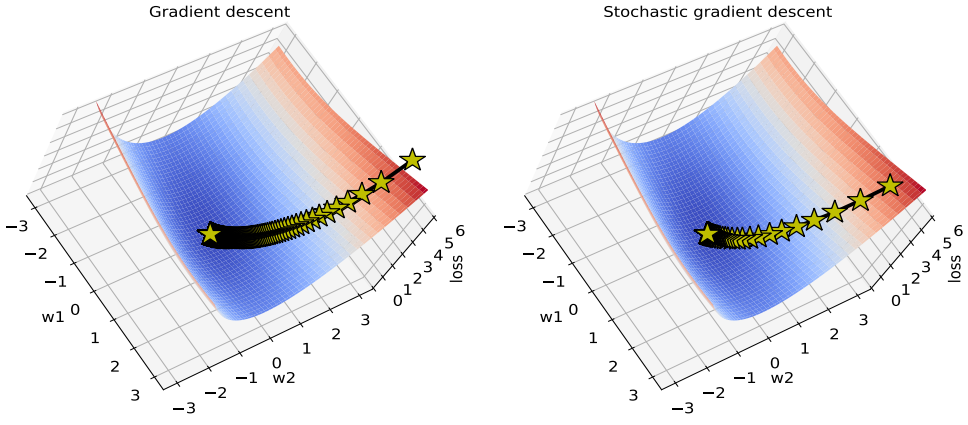

Рис. 2: Градиентные методы

- Видна сходимость в обоих случаях.
- При значении параметров по умолчанию шаг градиентного спуска достаточно велик, поэтому стохастический метод какое-то время «топтался» в окрестности минимума.
- При установлении параметров $\alpha = 0.1, \beta = 0.5$ длина шага уменьшилась, но увеличилось число шагов.
- Также заметно, что стохастическому методу во втором случае пришлось совершить существенно меньшее число шагов по сравнению с классическим градиентным спуском.

3.2 Подготовка данных

Убедившись, что градиентный спуск корректен, перейдем к решению задачи бинарной классификации эмоциональной окраски текстов.
Данные представляют собой тексты (52061 шт.) и соответствующие им метки 0 или 1, обозначающие нетоксичный и токсичный тексты соответственно.

You, sir, are my hero. Any chance you remember what page that's on?	0
from me as well, use the tools well.	0
COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1

Таблица 1: Данные

Тексты содержат спецсимволы, которые не видны при отображении. Заменим все символы кроме букв и цифр на пробелы, приведем к нижнему регистру и обрежем лишние пробелы. Изменим вектор меток под нашу задачу.

you sir are my hero any chance you remember what page that s on	-1
congratulations from me as well use the tools well talk	-1
cocksucker before you piss around on my work	1

Таблица 2: Данные

Совершили предварительную подготовку, теперь можно переходить к векторизации.
С помощью sklearn.feature_extraction.text.CountVectorizer преобразуем тексты в разряженную матрицу путем подсчета встречаемых в тексте слов. Будем включать только те слова, которые вошли хотя бы в 1% всех текстов.
Обозначим полученную матрицу — $X \in \mathbb{R}^{52061 \times 568}$, ответы — $y \in \{-1, 1\}^{52061}$.

3.3 О подсчете градиента

Так как производная — это предел приращения функции к приращению аргумента при стремлении приращения аргумента к нулю, можно попробовать считать производную численно.

$$f'(x) = lim_{\varepsilon \to 0} \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$

Сравним данный метод подсчета производной с использованием готовой аналитически посчитанной функции при вычислении градиента бинарной логистической функции потерь на выборке X с ответами на ней y в точке $w^* = (1_0, \ldots, 1_{568})$.

$\nabla_w L(w^*, X, y)$	1	2	...	567	568	Время
Числ. метод	0.01396074	1.01037472	...	1.2590192	1.02008926	6.03с
Аналитич. метод	0.01395664	1.01037171	...	1.25872622	1.02008507	64.1мс

Таблица 3: Градиент

Расхождение в точности начинается с 5-6 знака, но разница во времени подсчета слишком существенна. Поэтому используем аналитический метод. Дополнительно убедились, что аналитически найденная функция градиента соответствует градиенту логистической бинарной функции потерь с учетом $l2$-регуляризации.

3.4 Подбор параметров

Исследуем поведение модели в зависимости от гиперпараметров и выбранного метода градиентного спуска.

3.4.1 Классический градиентный спуск (GD)

Запустим перебор по сетке параметров $\eta = \frac{\alpha}{k^\beta}$, $\alpha, \beta > 0$. Оставим все параметры по умолчанию, критерием останова установим достижение максимального числа итераций (по умолчанию 1000).

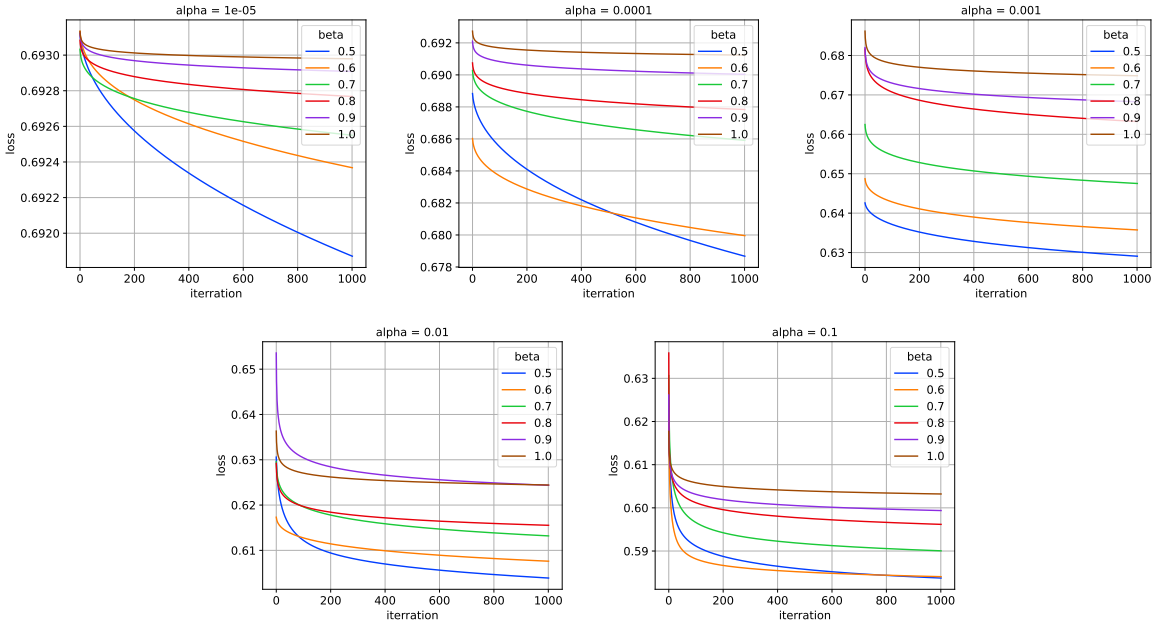

Рис. 3: Значение функции потерь от номера итерации и гиперпараметров шага обучения

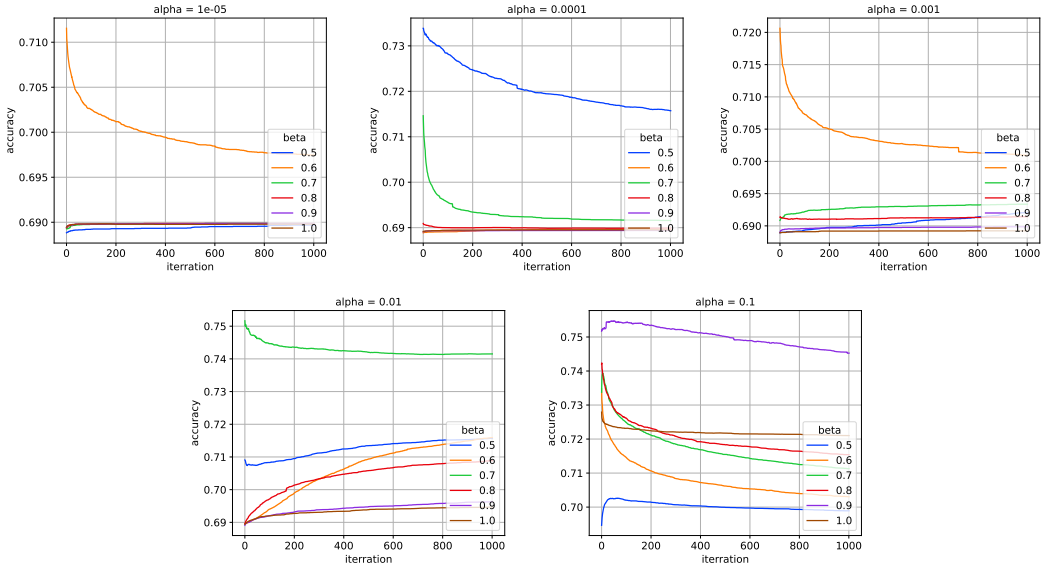

Рис. 4: Значение точности (accuracy) от номера итерации и гиперпараметров шага обучения

Получаем не самый очевидный результат. С ростом числа итераций уменьшается целевая функция, но при этом уменьшается и точность классификатора (причину см. в [3.4.4]). Возможно, проблема заключается в интерпретации вероятностей, которые возвращает модель, потому что модель обучается не улучшать accuracy - она обучается минимизировать целевую функцию. С этим она справляется «лучше» (по значению целевой функции) и «быстрее» (по количеству необходимых итераций) при параметрах $\alpha = 0.1$ и $\beta = 0.6$

Найдем оптимальное начальное приближение. Как было перечисленно в [2.5.2] рассмотрим:

- $w = (0, \dots, 0)$
- $w \sim N(0, 1)$
- $w \sim U(-1, 1)$
- $w = \left(\frac{\langle f_j, y \rangle}{\langle f_j, f_j \rangle}\right)_{j=1}^{d}, w_0 = 0$, где y - столбец меток, f_j - столбец j-ого признака.
- Обучение по 100 случайным объектам с нулевой начальной инициализацией (на графиках - default).

Фиксируем число максимальных итераций 400. За критерий останова возьмем следующее требование $|Q_{k+1} - Q_k| < \varepsilon$, где $\varepsilon = 10^{-5}$, Q_k - значение целевой функции на k-ой итерации.
α и β зафиксируем из [3.4.1] равными 0.1 и 0.6 соответственно. Можно заметить, что веса зажаты

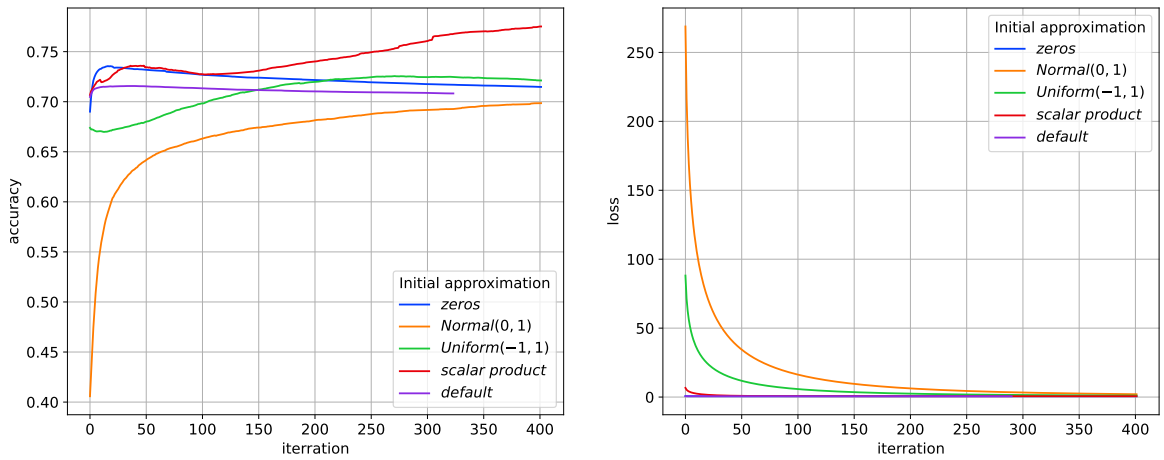

Рис. 5: Значение точности (accuracy) и функции потерь от номера итерации и начального приближения

в малой окрестности 0. Так как, при добавлении случайности, по модулю не превосходящей 1, функция потерь приобретает достаточно большие значения. А те приближения, которые сразу легли в «малую» окрестность 0, имеют ошибку на порядки меньше. Сравнив графики точности и целевой функции, выберем метод инициализации путем скалярных произведений для лучшей accuracy. Также стоит отметить единственный метод, который сошелся быстрее, чем за 400 итераций — метод обучения начального приближения на 100 случайных объектах.

3.4.2 Стохастический градиентный спуск (SGD)

Запустим перебор по сетке параметров $\eta = \frac{\alpha}{k^\beta}$, $\alpha, \beta > 0$. Оставим все параметры по умолчанию, критерием останова установим достижение максимального числа итераций 100, замерять значения целевой функции и accuracy будем каждый раз, когда была пройдена $\frac{1}{10}$ часть выборки, размер батча зафиксируем равным 128.

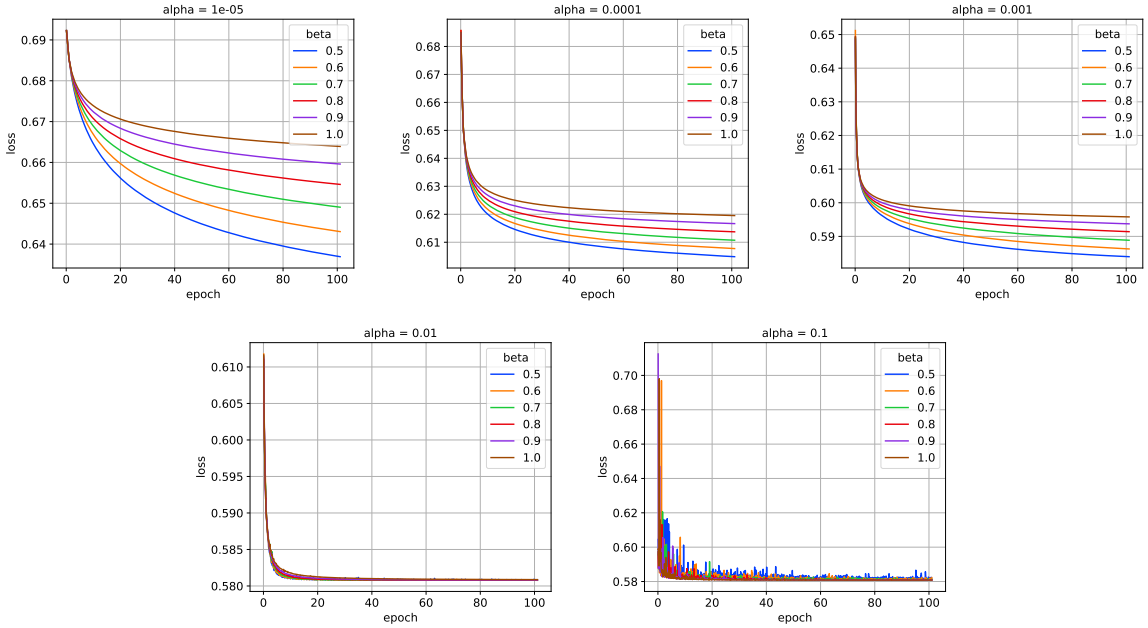

Рис. 6: Значение функции потерь от номера итерации и гиперпараметров шага обучения

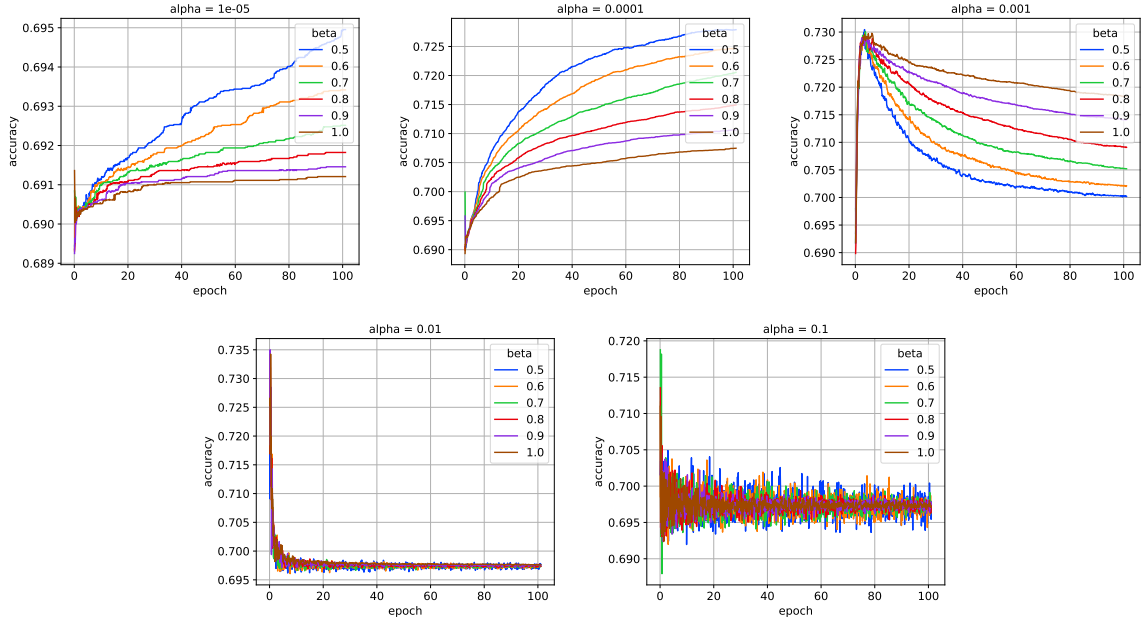

Рис. 7: Значение точности (accuracy) от номера эпохи и гиперпараметров шага обучения

Наблюдаем аналогичную картину — местами с ростом итераций уменьшается как точность, так и значение целевой функции. Нетрудно видеть, каким «шумным» стало изменение целевой функции и точности в связи с добавлением в алгоритм приближения градиента по случайной под-

выборке фиксированного размера. Причем при больших значениях длины шага «шумы» тоже становятся сильнее.
Таким образом, достаточно «быстрый» и «гладкий» спуск функции потерь показывают параметры $\alpha = 0.01$ и β - любое из перечисленных. Для определенности зафиксируем $\beta = 1$.

Переберем начальные приближения.
Фиксируем число максимальных итераций 40, замерять значения целевой функции и accuracy будем каждый раз, когда была пройдена $\frac{1}{10}$ часть выборки. Критерий останова - $|Q_{k+1} - Q_k| < \varepsilon$, где $\varepsilon = 10^{-5}$
α и β фиксируем из [3.4.2], размер батча - 128.

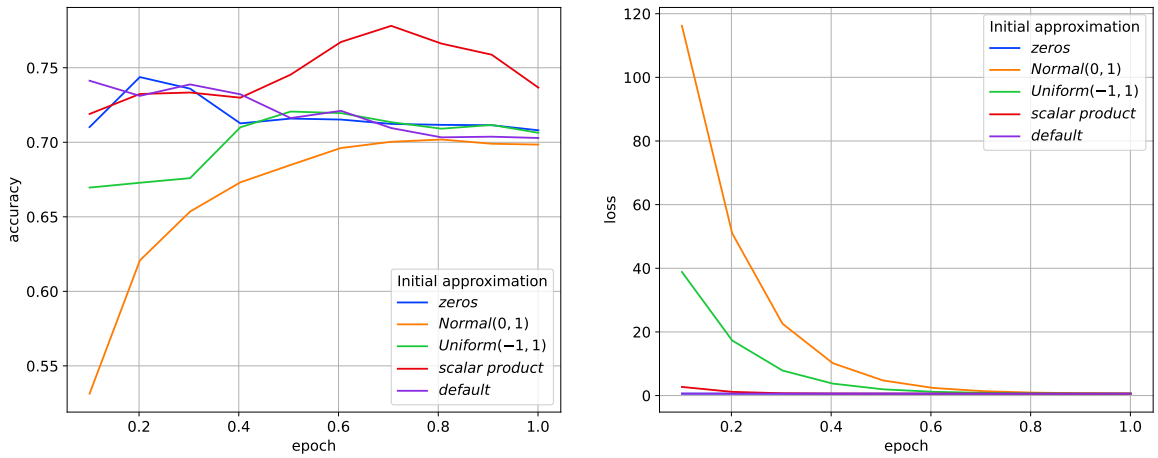

Рис. 8: Значение точности (accuracy) и функции потерь от номера эпохи и начального приближения

Получаем аналогичные обычному градиентному спуску тенденции поведения графиков, но при этом начальная ошибка меньше. Это можно попробовать объяснить тем, что к моменту начала измерения было уже совершено определенное число шагов. Начальным приближением фиксируем скалярные произведения.

3.4.3 Дообучение оставшихся параметров

Исследуем два оставшихся параметра — коэффициент регуляризации и размер батча для стохастического градиентного спуска.

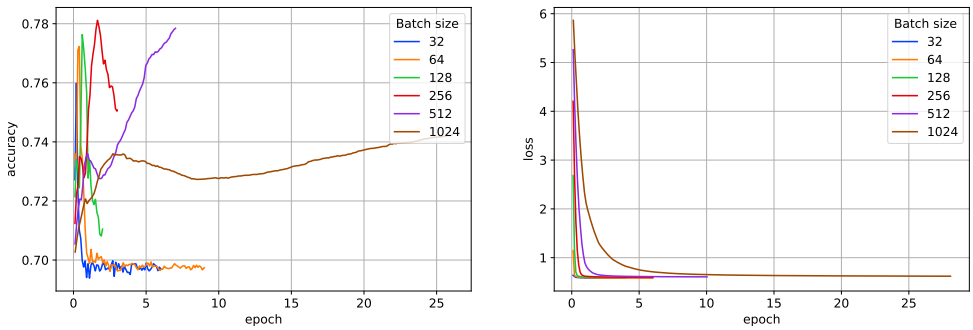

Рис. 9: Значение точности (accuracy) и функции потерь стохастического спуска от номера эпохи и размера батча

Достаточно быструю сходимость и «хорошую» accuracy дают батчи размеров 256 и 512. Для ускорения работы алгоритма остановимся на размере батча равного 256.

Для нахождения лучшего коэффициента регуляризации используем:

- $\alpha = 0.01, \beta = 1$
- $w = \left(\frac{\langle f_j, y\rangle}{\langle f_j, f_j\rangle}\right)_{j=1}^{d}, w_0 = 0$
- $batch\ size = 256$
- Критерий останова — $|Q_{k+1} - Q_k| < 10^{-5}$
- Замеряем целевую функцию и accuracy при проходе $\frac{1}{10}$ части выборки

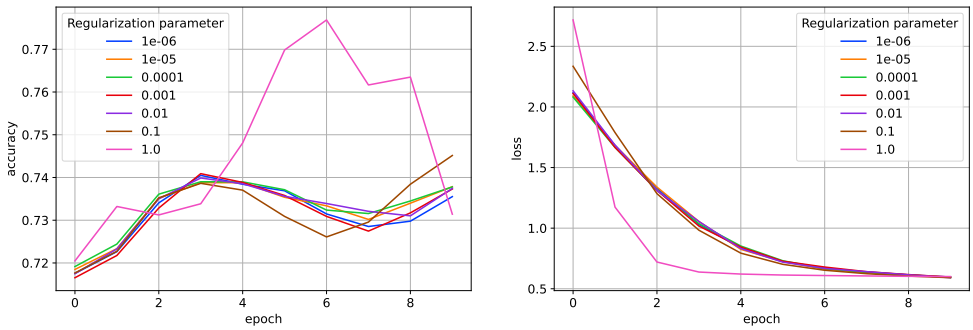

Рис. 10: Значение точности (accuracy) и функции потерь стохастического спуска от номера эпохи и размера батча

Лучшее качество и более быструю скорость сходимости дает коэффициент $\lambda = 1$.

3.4.4 Стоит ли обучать смещение (bias)?

Рассмотрим обычный и стохастический градиентные спуски с лучшими параметрами. Ограничимся 100 итерациями, для стохастического градиентного спуска под итерацией будем подразумевать 1 проход по $\frac{1}{10}$ части выборки.

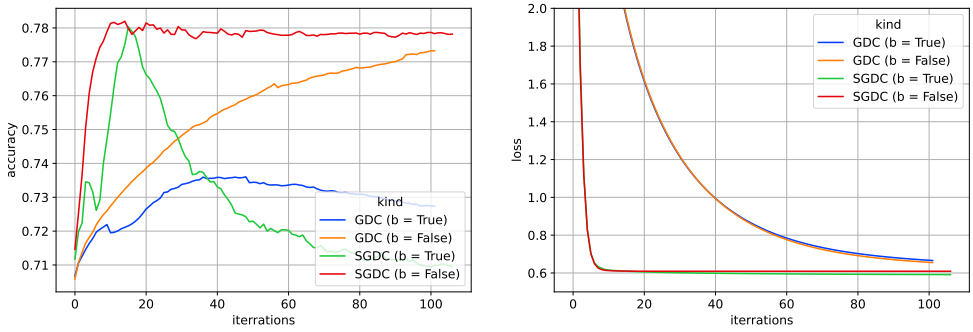

Рис. 11: Значение точности (accuracy) и функции потерь от используемого вида градиентного спуска и наличия коэффициента смещения

По итогу видно, что обучение смещения в данной задаче не дает сильного улучшения скорости обучения, но при этом с числом итераций уменьшается accuracy модели. Это может быть связано с тем, что критерий останова достигается быстрее, чем полностью обучится bias. В связи с полученными наблюдениями в данной задаче не будем обучать смещение.

3.4.5 Как интерпретировать вероятности?

Зафиксируем «лучшую» модель на основе стохастического градиентного спуска.
Исследуем, как выбор порога, по которому принимается решение отнести объект к положительному или отрицательному классу, влияет на точность модели.

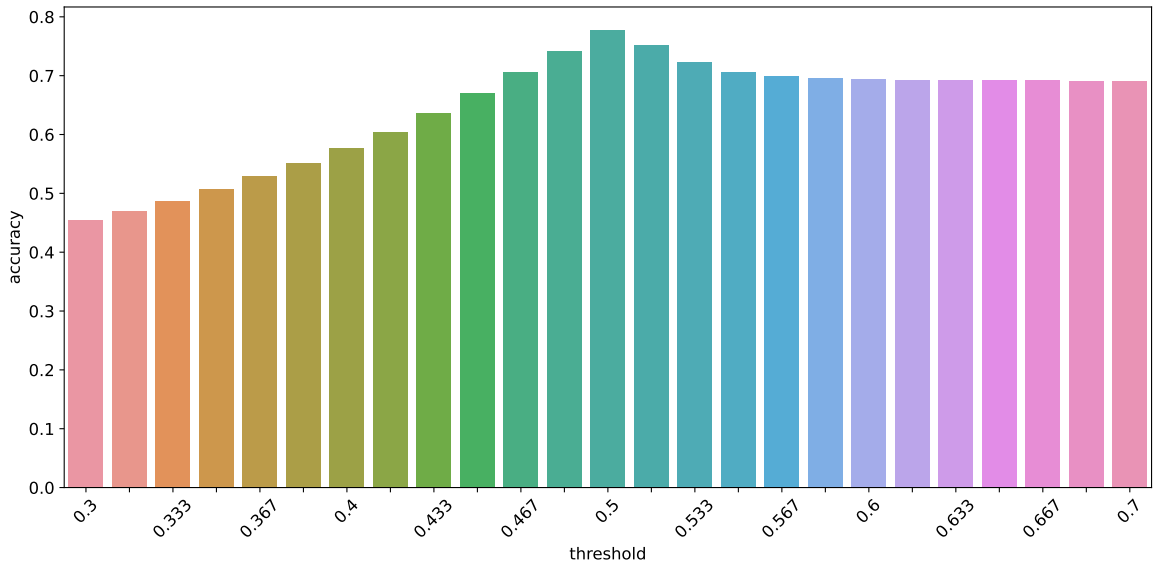

Рис. 12: Значение точности (accuracy) от порога классификации

Наилучшее качество достигается при значении порога 0.5, то есть интуитивное представление о вероятностях, которые должна выдавать модель, оправдалось.

3.4.6 Итог

Резюмируем полученные результаты: фиксируем «лучшие» параметры из разделов [3.4.1] ÷ [3.4.4]. Критерий останова — $|Q_{k+1} - Q_k| < 10^{-5}$, максимально число итераций — по умолчанию.
Обучаем модели и замеряем качество на тестовой выборке.

Метод	Точность	Время
GD	0.763687	36.9c
SGD	0.763687	5.28c

Таблица 4: Итоговое сравнение

При одинаковых значениях точности разница во времени составила 31 секунду (под временем понимается полное время обучения и предсказаний). То есть теоретические представления о различии в скорости работы обычного градиентного спуска и стохастического оправдались.

3.5 Улучшение качества данных

Попробуем улучшить качество модели с помощью дополнительной обработки текстов.
Для этого применим такие методы как *лемматизация* и *удаление стоп-слов*.
Под *лемматизацией* понимается приведение слова к его начальной форме (*лемме*).
Стоп-словами называют дополнительные слова, не несущие смысловой нагрузки.
Для лемматизации будем использовать nltk.stem.WordNetLemmatizer, а список стоп-слов возьмем из nltk.corpus.stopwords.

Список стоп слов:
i, me, my, myself, we, our, ours, ourselves, you, you're, you've, you'll, you'd, your, yours, yourself, yourselves, he, him, his, himself, she, she's, her, hers, herself, it, it's, its, itself, they, them, their, theirs, themselves, what, which, who, whom, this, that, that'll, these, those, am, is, are, was, were, be, been, being, have, has, had, having, do, does, did, doing, a, an, the, and, but, if, or, because, as, until, while, of, at, by, for, with, about, against, between, into, through, during, before, after, above, below, to, from, up, down, in, out, on, off, over, under, again, further, then, once, here, there, when, where, why, how, all, any, both, each, few, more, most, other, some, such, no, nor, not, only, own, same, so, than, too, very, s, t, can, will, just, don, don't, should, should've, now, d, ll, m, o, re, ve, y, ain, aren, aren't, couldn, couldn't, didn, didn't, doesn, doesn't, hadn, hadn't, hasn, hasn't, haven, haven't, isn, isn't, ma, mightn, mightn't, mustn, mustn't, needn, needn't, shan, shan't, shouldn, shouldn't, wasn, wasn't, weren, weren't, won, won't, wouldn, wouldn't.

Было	Стало
you sir are my hero any chance you remember what page ...	sir hero chance remember page
congratulations from me as well use the tools well talk	congratulation well use tool well talk
cocksucker before you piss around on my work	cocksucker piss around work

Таблица 5: Данные

Применяем тот же метод кодирования sklearn.feature_extraction.text.CountVectorizer с условием, включения слов, вошедших более чем в 1% текстов. Все параметры оставляем «лучшими» и находим «лучшее» начальное приближение для новой выборки (см. [3.4]).

	GD(до)	GD(после)	SGD(до)	SGD(после)
Точность	0.763687	0.797495	0.763687	0.794738
Время	36.9c	8.96c	5.28c	1.61c
Кол-во признаков	568	456	568	456

Таблица 6: Сравнение алгоритмов с доп. обработкой текстов и без нее

Наблюдается уменьшение размерности признакового пространства и, как следствие, уменьшение времени работы алгоритма. При этом точность улучшается на $\sim 3\%$. Также можно заметить, что в условиях данной задачи и с подобным образом подобранными гиперпараметрами SGD второй раз показывает уменьшение во времени работы по сравнению с GD приблизительно в 3-4 раза.

3.6 Гиперпараметры отбора признаков

В предыдущем разделе [3.5] использовался метод подсчета слов *Bag of words*, но также можно учитывать и частоту встречаемости слова по всем текстам, для реализации этого метода воспользуемся sklearn.feature_extraction.text.TfidfVectorizer.
Параметрами для векторизаций служат min df и max df — минимальная и максимальная частоты, с которыми слово может появляться в документах. Если частота встречаемости слова не удовлетворяет данным параметрам, то слово пропускается.
Исследуем два данных подхода и переберем параметры на основе «лучшей» SGD модели. Для упрощения работы начальное приближение будем брать нулевым.
Исследовать будем следующим образом: отберем «оптимальный» min df и далее, фиксировав выбранный параметр выберем max df.

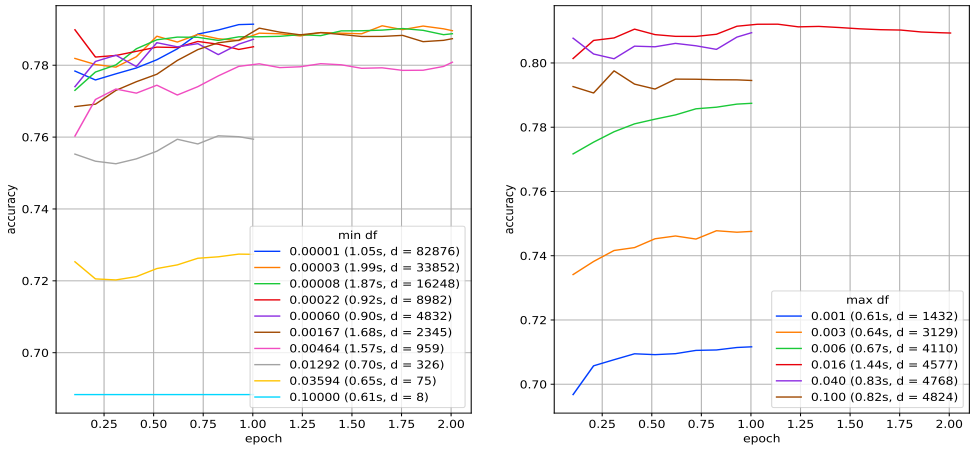

Рис. 13: Значение точности (accuracy) от параметров min df и max df при кодировании с помощью Bag of words

На первом графике при размерности признакового пространства меньше 1000 наблюдается уменьшение точности. Значит рассматриваем варианты размерности больше 1000. В таком диапазоне выделяются значения параметра $3 \cdot 10^{-5}, 8 \cdot 10^{-5}, 167 \cdot 10^{-5}$. Для их сходимости потребовалось совершить 2 полных обхода по выборке $\Rightarrow$ затратить в 2 раза больше времени на выполнение по сравнению с остальными методами. Из оставшихся значений выберем то, которое обладает минимальной размерностью признакового пространства (для более быстрой работы алгоритма). Таким образом, min df $= 6 \cdot 10^{-4}$.
Зафиксировав значение минимальной частоты, перебираем значения максимальной частоты. Для лучшей тончости алгоритма рассматриваем два возможных значения 0.016 и 0.4. Из них более быструю сходимость показал max df $= 0.04$.
Важно отметить, что точность при добавлении оптимального параметра max df выросла с ~ 0.79 до ~ 0.81.

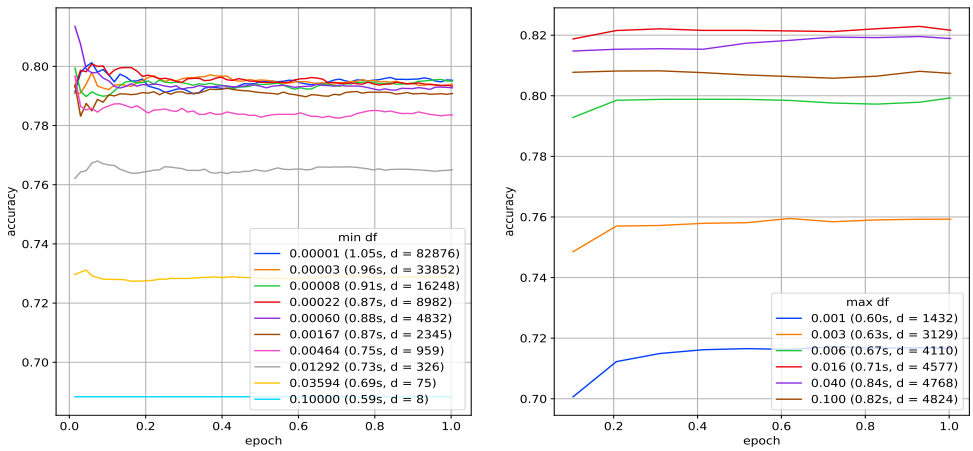

Рис. 14: Значение точности (accuracy) от параметров min df и max df при кодировании с помощью tf-idf

Можно провести полностью аналогичные рассуждения с предыдущим методом кодирования, только теперь при всех значениях параметров сходимость происходит за 1 эпоху. То есть большее

число параметров являются «оптимальными», так как большая их часть показывает относительно похожие высокие результаты. В таком случае возьмем значения аналогичное тому, что было в предыдущем случае, так как оно не только является одним из «оптимальных», но и дополнительно позволит произвести более объективное сравнение двух методов векторного представления данных.
Зафиксировав min df $= 6 \cdot 10^{-4}$, перебираем значения параметра max df, аналогично тому, как это было сделано выше. Лучшим по соотношению качество-время выступает значение max df $=$ 0.016, что не совпадает с выбором оптимального max df при векторизации методом подсчета.

Итого обучаем модели на базе SGD с «лучшими» параметрами, «лучшим» начальным приближением и использованием «лучших» параметров векторизаций. Замеряем время обучения и предсказаний, считаем точность на тестовой выборке.

	Точность	Время
Bag of words	0.833188	2.51с
Tf-idf	0.823612	801мс

Таблица 7: Сравнение алгоритмов с разными способами векторизации текстов

Однозначно определить, какой метод лучше, в данном случае не просто. Разность точности в 1% достаточно существенна, но при этом при использовании Tf-idf кодировки время работы в 3 раза меньше по сравнению с методом подсчета.

3.7 Анализ ошибок

Возьмем модель с наибольшей точностью из [3.6]. Посмотрим на допускаемые ей ошибки на обучающей выборке.

Ложно положительно классифицированные тексты:

- lmao what a n00b go and listen to manele
- know the sex of the foetus
- hate is my topic is hate if you hate a person means you like that person your hate can turn into your love
- is jeff garcia gay or not
- cnn says mother died it s on their home page
- that is to say are you gay or straight
- did you poop in your pants

Можно предположить, что модель обращает внимание на слова, которые в языке имеют негативную окраску, но при этом модель не смотрит на контекст их появления. К примеру, в данном случае слова «lmao», «n00b», «sex», «hate», «gay», «die», «poop» можно отнести к «негативным», так как нередко они могут использоваться в подобном контексте.

Ложно отрицательно классифицированные тексты:

- you should be fired you re a moronic wimp who is too lazy to do research it makes me sick that people like you exist in this world
- kill all niggers i have hard that others have said this should this be included that racists sometimes say these
- fuck off you are not an administrator you don t have the authority to tell me what to do
- fuck you fuck you award go fuck yourself

- matt hardy is so fucky italic text media example ogg matt hardy is so fucky
- i don t care what you say here i don t believe one sentence anymore
- you sir are an imbecile and a pervert

Можно заметить, что модель получилась устойчивой к расизму и слову «fuck» во всех его формах. Наверное, это можно объяснить тем, что слово «fuck» может использоваться по поводу и без, поэтому модель однозначно не классифицировала его как индикатор положительного класса. Еще можно заметить, что модель не может распознавать «пассивную-агрессию», то есть высказывания, в которых нет прямых оскорблений или они не так явно читаются.

Итого: проанализировав ошибки, можно предположить, что модель обучилась классифицировать тексты по наличию в них особых слов-триггеров. Возможно, для дальнейшего улучшения качества работы модели стоит обучить ее распознавать контекст (к примеру, попробовать добавить n-граммы).

4 Заключение

В ходе данного исследования были установлены следующие результаты:

1. Качество обучаемой модели напрямую зависит от параметров обучения (длина шага обучения, коэффициент регуляризации, начальное приближение, размер батча).
2. На данном наборе данных стохастический градиентный спуск показал сходимость в среднем в 3-4 раза быстрее по сравнению с классическим градиентным спуском.
3. Вероятности логистической регрессии интерпретируются интуитивно, то есть оптимальный порог для максимальной accuracy равен 0.5.
4. Лемматизация и удаление стоп-слов — полезный инструмент для предобработки текстов, который не только повышает качество итоговой модели, но и сокращает размерность признакового пространства.
5. Существует несколько методов векторизации текстов (были рассмотрены Bag of words и tf-idf), и однозначно утверждать, какой из методов лучше, нельзя даже на примере данного датасета. В одном лучше точность, в другом — время выполнения.
6. Модель, обученная на векторах из закодированных слов, не может получать информацию из контекста. Для решения этой проблемы необходимо использовать дополнительные эвристики, которые не были рассмотрены в данной работе.

5 Источники

[1] Школа анализа данных «Учебник по машинному обучению».
[2] «Машинное обучение 1» ФКН ВШЭ, Лектор: Соколов Евгений Андреевич.
[3] «Технологическая практика» ММП ВМК семинарские занятия.