Задание 2. Применение линейных моделей для анализа токсичности комментария

Сумина Евгения, практикум 317 группы

2020

Введение

В задании требовалось ознакомиться с линейными моделями и градиентными методами обучения, провести сравнительный анализ градиентного спуска и стохастического градиентного спуска. Также было необходимо провести теоретические рассчеты формулы градиента для задач бинарной и мультиномиальной логистической регрессии. Предлагалось ознакомиться с некоторыми методами, применяющимися при анализе текстов (лемматизация, BagOfWords, Tfidf).

Теоретические рассчеты

Функция потерь для задачи бинарной логистической регрессии выглядит так:

$$Q(w) = \frac{1}{L}\sum_{i=1}^{L}\log\left[1+\exp(-y_i\langle w, x_i\rangle)\right] + \frac{C}{2}w^2$$

Выведем формулу градиента.

$$\nabla_w Q(w) = \frac{1}{L}\nabla_w \sum_{i=1}^{L}\log\left[1+\exp(-y_i\langle w, x_i\rangle)\right] + \frac{C}{2}\nabla_w w^2 =$$

$$= \frac{1}{L}\sum_{i=1}^{L}\frac{-y_i x_i \exp^{-y_i\langle w, x_i\rangle}}{1+\exp^{-y_i\langle w, x_i\rangle}} + Cw$$

Градиентом является вектор длины D, где D - количество признаков у объекта.

Теперь рассмотрим вывод градиента функции потерь для задачи мультиномиальной регрессии. Будем оптимизировать функционал

$$Q(w) = -\frac{1}{L}\sum_{i=1}^{L}\log(P(y_i|x_i)) + \frac{C}{2}\sum_{i=1}^{D}\|w_i\|^2$$

При этом вероятность $P(y_i|x_i) = P(y = y_i|x_i) = \frac{\exp^{\langle w_{y_i}, x_i\rangle}}{\sum_{k=1}^{D}\exp^{\langle w_k, x_i\rangle}}$

Подставим значение вероятности в функцию потерь:

$$Q(w) = -\frac{1}{L}\sum_{i=1}^{L}\log(P(y_i|x_i)) + \frac{C}{2}\sum_{i=1}^{D}\|w_i\|^2 = \frac{1}{L}\sum_{i=1}^{L}\log(\frac{\sum_{k=1}^{D}\exp^{\langle w_k, x_i\rangle}}{\exp^{\langle w_{y_i}, x_i\rangle}}) + \frac{C}{2}\sum_{i=1}^{D}\|w_i\|^2 =$$

$$Q(w) = \frac{1}{L}\sum_{i=1}^{L}\left[\log(\sum_{k=1}^{D}\exp^{\langle w_k, x_i\rangle}) - \langle w_{y_i}, x_i\rangle\right] + \frac{C}{2}\sum_{i=1}^{D}\|w_i\|^2$$

Теперь продифференцируем по w_{pq}:

$$\frac{\partial Q(w)}{\partial w_{pq}} = \frac{1}{L}\sum_{i=1}^{L}\frac{x_{iq}\exp^{\langle w_p, x_i\rangle}}{\sum_{k=1}^{D}\exp^{\langle w_k, x_i\rangle}} - \frac{1}{L}\sum_{i=1}^{L}x_{iq}\left[y_i = p\right] + Cw_{pq}$$

Градиентом является матрица размера $n \times D$, где n - количество классов, D - количество признаков.

Покажем, что при количестве классов = 2, задача мультиномиальной логистической регрессии сводится к бинарной логистической регрессии. Воспользуемся промежуточным выражением, полученным на прошлом этапе.

$$Q(w) = \frac{1}{L}\sum_{i=1}^{L}\log(\frac{\sum_{k=1}^{D}\exp^{\langle w_k, x_i\rangle}}{\exp^{\langle w_{y_i}, x_i\rangle}}) + \frac{C}{2}\sum_{i=1}^{D}\|w_i\|^2 =$$

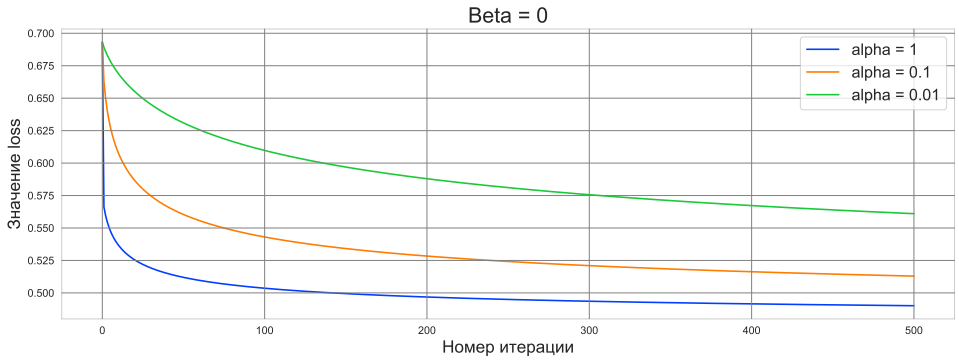
$$\frac{1}{L}\sum_{i=1}^{L}\log(1+\exp^{\langle w_k - w_{y_i}, x_i\rangle}) + \frac{C}{2}\sum_{i=1}^{D}\|w_i\|^2 = \frac{1}{L}\sum_{i=1}^{L}\log(1+\exp^{\langle w_k, x_i\rangle - \langle w_{y_i}, x_i\rangle}) + \frac{C}{2}\sum_{i=1}^{D}\|w_i\|^2 =$$

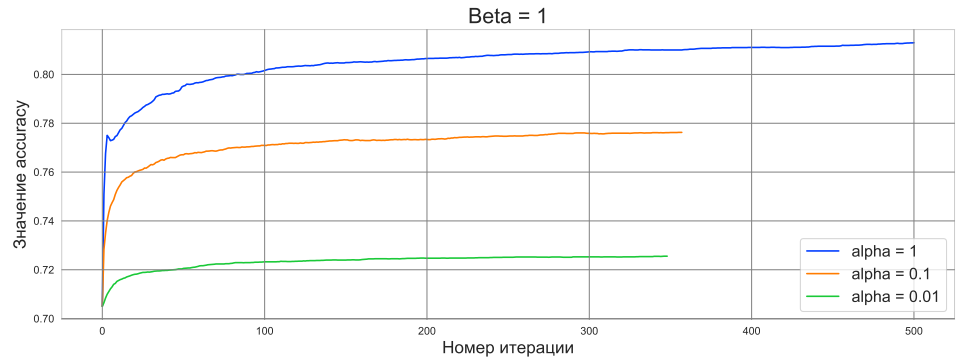
$$\frac{1}{L}\sum_{i=1}^{L}\log(1+\exp^{-y_i\langle w, x_i\rangle}) + \frac{C}{2}\sum_{i=1}^{D}\|w_i\|^2$$

Первый эксперимент

Все текстовые данные были преобразованы в единый формат: все символы были приведены к нижнему регистру, символы, не являющиеся буквами и цифрами, были заменены на пробелы. С помощью CountVectorizer выборка была преобразована в разреженную матрицу, при этом был выбран параметр $min_df = 0.0001$. В результате были отобраны 16950 признаков. К тому же, было решено разбить тренировочные данные на train и валидацию, все значения accuracy предсказывались на валидационных данных.

Рассмотрим поведение градиентного спуска при различных значениях параметров α, β и начального приближения. Для каждого фиксированного β из массива возможных значений $[0, 0.5, 1]$ будем строить зависимость функции потерь для различных α.Было принято решение строить отдельный график для каждого значения β для более удобного анализа.



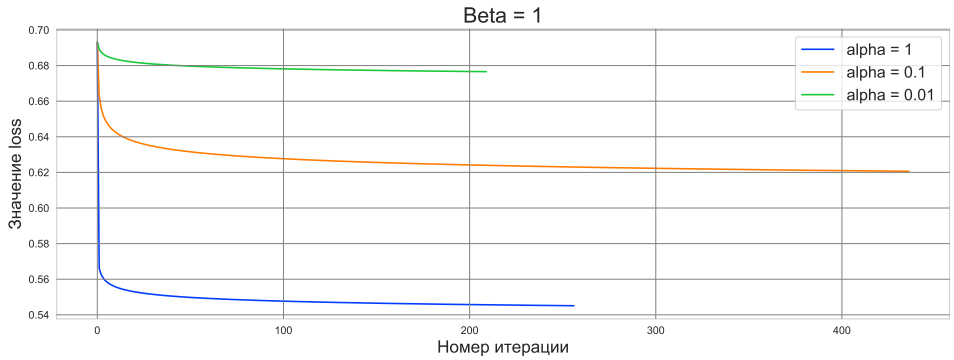


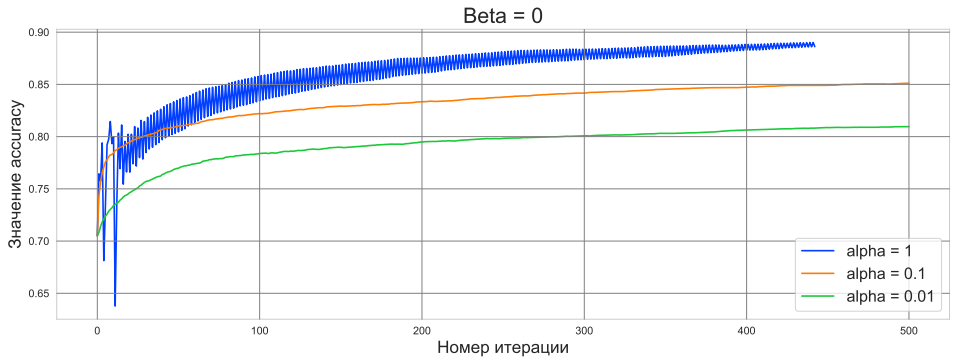
Можно заметить, что на всех графиках лучшие показатели достигаются при $\alpha = 1$. При этом при $\beta = 1$ метод сходится быстрее, чем пройдет максимальное число итераций. Это объясняется тем, что при маленьком значении шага (β находится в знаменателе, и с его увеличением шаг, соответственно, уменьшается) значение loss меняется меньше и быстрее достигает признака остановки. Думаю, стоит проанализировать, как выглядит функция потерь для различных β при фиксированном $\alpha = 1$. Рассмотрим это чуть позже. Для начала рассмотрим поведение значения accuracy в зависимости от номера итерации.

При $\beta = 0$ и максимальном значении $\alpha = 1$ наблюдаются заметные скачки значения accuracy. При этом с увеличением номера итерации диаметр скачков уменьшается, и в целом градиент двигается в определенном направлении. Остальные функции выглядят более сглаженными. При $\beta = 1$ мы снова можем наблюдать сходимость градиентного спуска раньше, чем будет пройдено максимальное число итераций. На всех трех графиках лучшее качество достигается при $\alpha = 1$.

Теперь зафиксируем $\alpha = 1$, т.к. этот показатель давал лучшие результаты на всех экспериментах, и рассмотрим, как меняется качество при изменении значения β.

Для $\beta = 0$ наблюдаются заметные скачки, особенно большой разброс в самом начале градиентного спуска. Также стоит заметить, что при $\beta = 0$ шаг градиентного спуска является константным значением: из-за этого на определенном шаге мы можем перепрыгнуть минимум. Поэтому для дальнейшего рассмотрения выберем $\beta = 0.5$. Видно, что при 500 итерациях градиентный спуск не достиг предела своего





обучения.

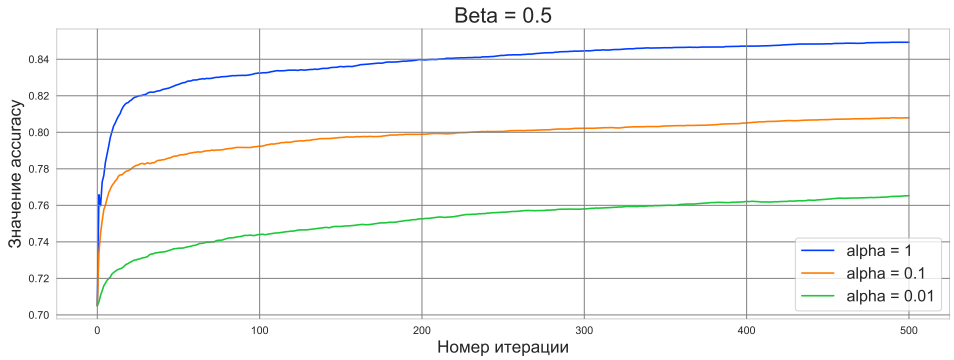
Теперь рассмотрим способы выбрать начальное приближение для градиентного спуска. Были предложены две стратегии: инициализирование вектора весов нулями либо набором случайных значений в промежутке от -1 до 1. При инициализировании нулями сходимость достигается быстрее. При этом на начальных итерациях наблюдаются довольно значительные скачки значений. Вероятно, нужный нам минимум находится в окрестности нулевого вектора, поэтому выберем в качестве инициализации его.

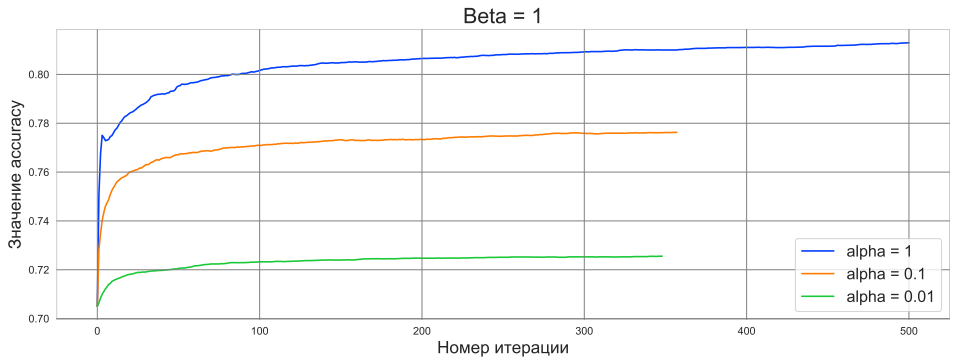
Второй эксперимент

Теперь проанализируем поведение стохастического градиентного спуска. Его отличие от градиентного спуска в том, что на каждой итерации градиент берется не по всей выборке, а по подмножеству объектов (батчу). За счет этого добиваются ускорения работы метода. К тому же, так повышается вероятность попасть в глобальный минимум: в случае обычного градиентного спуска велика вероятность остаться в локальном.

Так же, как и в предыдущем эксперименте, будем последовательно фиксировать значение β в диапазоне $[0, 0.5, 1]$ и смотреть, как меняется качество для различных α. Для начала, проанализируем поведение loss-функции:

Прежде всего, стоит заметить, что графики, в отличие от предыдущего эксперимента, не являются гладкими ни для каких значений α и β. Это объясняется





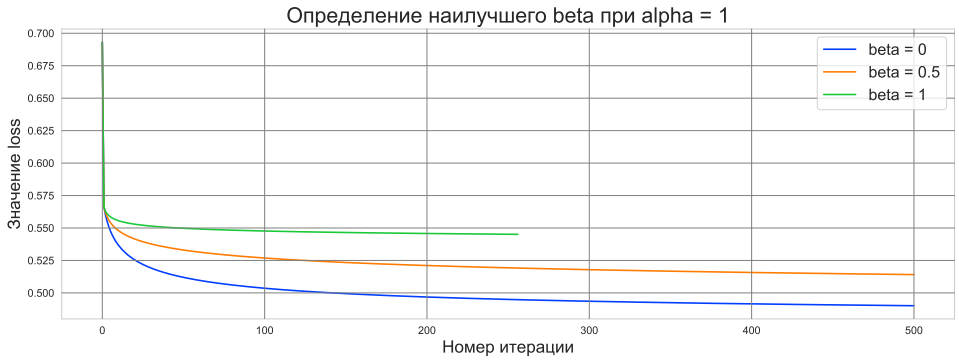
методикой построения стохастического градиентного спуска: при подсчете градиента по различным подмножествам всей выборки направления промежуточных градиентов могут отличаться. Лучшие показатели опять достигаются при $\alpha = 1$. Стоит заметить, что при таком значении α и $\beta = 0$ скачки градиентного спуска слишком значительны.

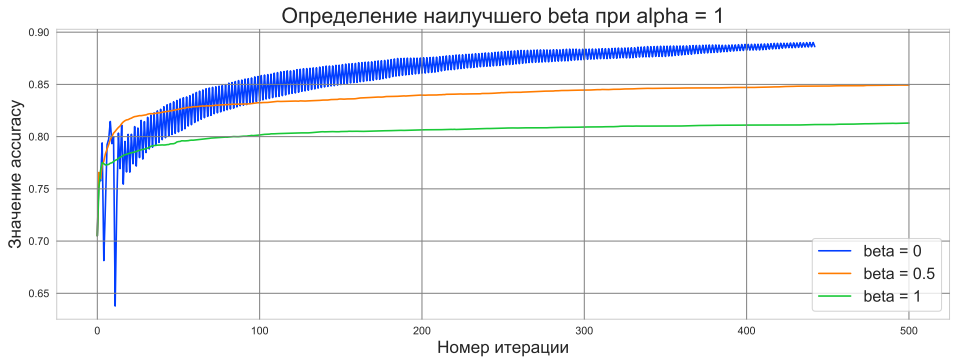
Теперь рассмотрим зависимость поведения accuracy от номера итерации.

При $\beta = 0$ метод сходится быстрее, чем будет превышено максимальное число эпох. Для $\alpha = 0$ наблюдаются скачки значений accuracy, но достигается лучшее качество. Для $\alpha = 0.1$ качество примерно такое же, но метод сходится дольше. С другой стороны, можно предположить, что при большем количестве итераций будет достигнуто лучшее качество. При остальных значениях параметра функция выглядит гладкой, причем с уменьшением размера шага гладкость увеличивается. Стоит отметить, что при $\beta = 0.51$ графики при различных α выглядят одинаково с точностью до смещения.

Т.к. лучшее качество достигается при $\alpha = 1$, зафиксируем этот параметр и проанализируем, как различные значения β отразятся на качестве.

При $\beta = 0$, как это было сказано ранее, наблюдаются значительные скачки значения accuracy. Также мной был построен график зависимости loss от номера итерации (который я не стала здесь приводить, т.к. он только подтверждает выводы, которые можно сделать при анализе accuracy), по которому можно заметить существенные колебания при таких значениях параметров. Лучшее качество (и возможность дальнейшего обучения) наблюдаются при $\beta = 0.5$, поэтому мы остановимся на





этом значении параметра.

Теперь рассмотрим различные варианты начального приближения. Ранее уже было выяснено, что нулевое приближение дает лучшее качество алгоритма. Посмотрим, подтвердится ли этот вывод для стохастического градиентного спуска.

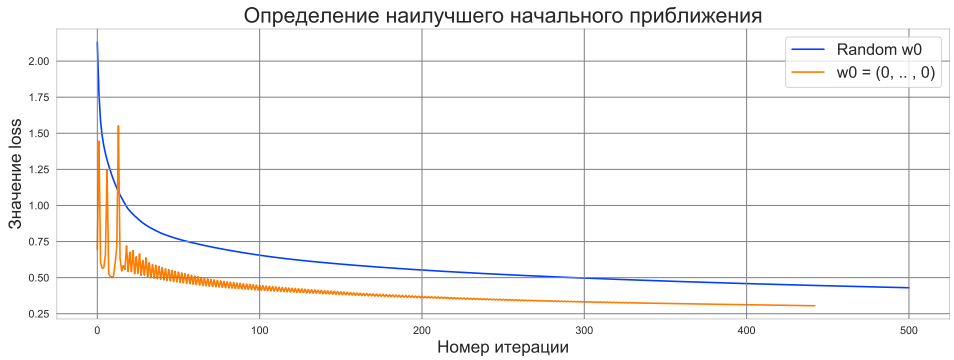
Видно, что произвольное задание весов снова не дает улучшения в сходимости алгоритма, и в целом при неправильном выборе начального приближения качество предсказания остается на примерно одном высоком уровне. Поэтому остановимся на инициализации начального вектора весов нулевыми значениями.

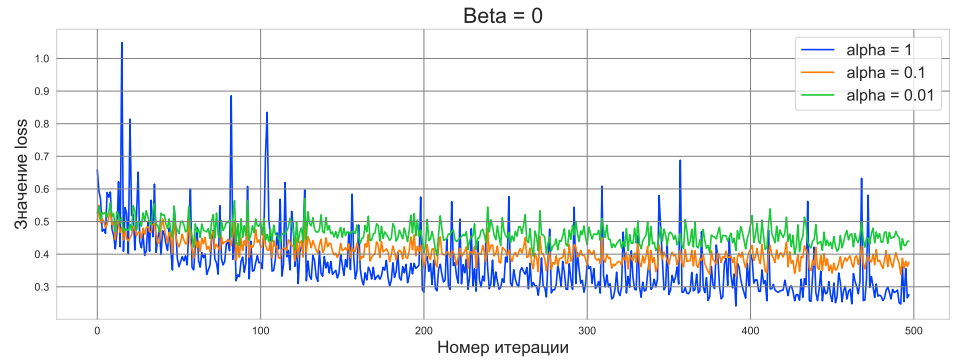
Наконец, выберем размер батча. Проанализируем скорость сходимости и значение accuracy при различных значениях параметра $batch_size$ из диапазона возможных значений $[100, 500, 1000, 1500]$.

Качество предсказания находится примерно на одном уровне для всех значений. При этом для $batch_size = 15001000$ сходимость достигается до превышения максимального числа итераций. В целом, при увеличении размера батча время, затрачиваемое на обучение метода, уменьшается. Исходя из предположения, что это не сильно влияет на качество обученной модели, выберем оптимальным размером батча для обучения $batch_size = 1000$.

Третий эксперимент

Проанализируем поведение алгоритмов с отобранными выше параметрами. Сравним показатели accuracy: прежде всего, заметим, что SGDClassifier быстрее, чем





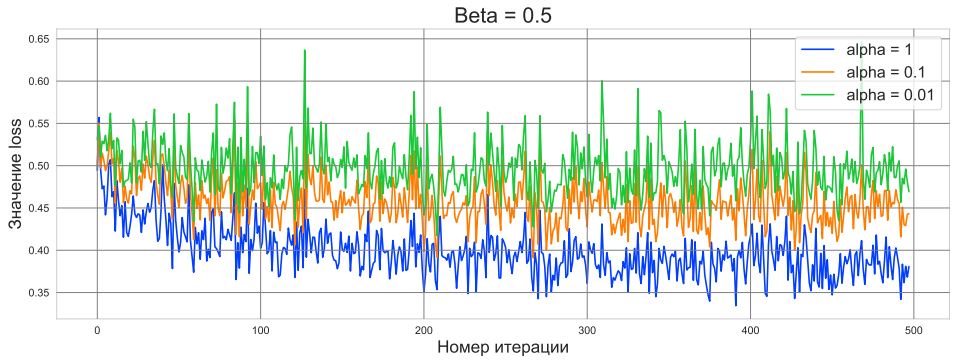
GDClassifier, достигает сравнительной точности, и за счет этого на равном числе итераций выдает лучшие показатели. Его график не такой гладкий, как у GDClassifier, что объясняется самим методом: градиент, подсчитанный по подмножеству, может менять свое направление, хотя в целом и двигается в нужную сторону.

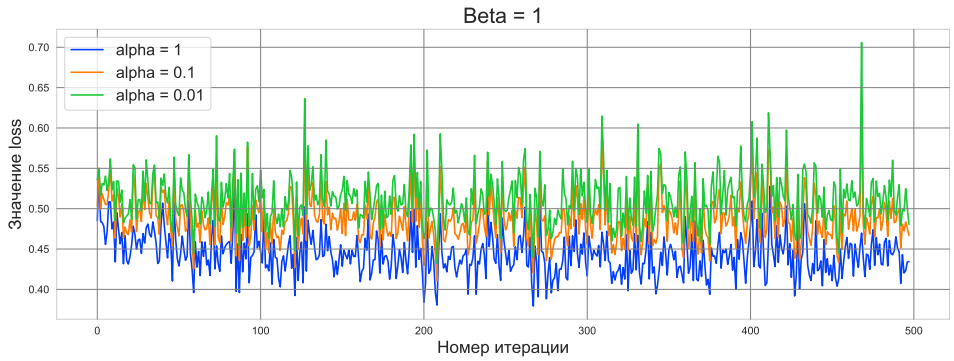
Значения loss выглядят интереснее: градиентный спуск гладко сходится к нужным показателям, при этом стохастический грудиентный спуск начинает с лучших значений, но за счет постоянных скачков разница в показателях с SGD небольшая. В целом, можем сделать вывод, что использование стохастического градиентного спуска выглядит более оправданным, т.к. его показатели качества значительно превосходят GDClassifier.

Четвертый эксперимент

Применим алгоритм лемматизации с помощью функции WordNetLemmatizer из библиотеки nltk. Удалим из текста стоп слова из словаря "english". В результате проведенных преобразований удалось понизить размерность признакового пространства с 16050 до 14305 признаков. Это произошло за счет того, что из текста были удалены незначащие слова (такие как предлоги и союзы), а однокоренные слова и одинаковые слова в разных формах стали учитываться как равные. Проверим, как это скажется на качестве и времени классификации. Значение accuracy будем считать на валидационной выборке, а обучаться на train.

После применения лемматизации качество предсказания повысилось, а вре-



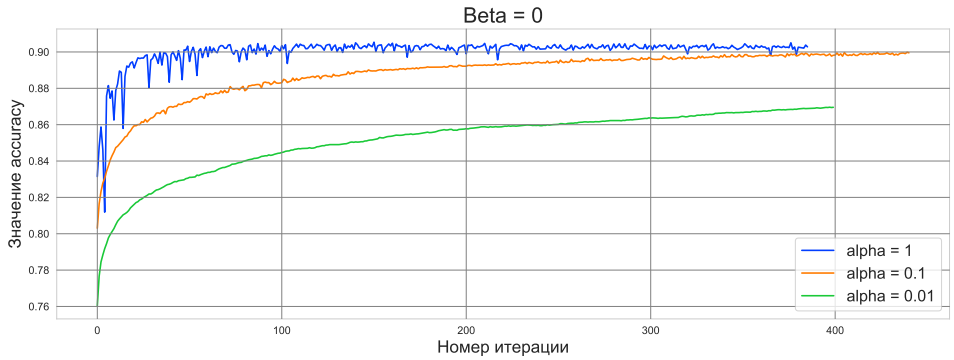


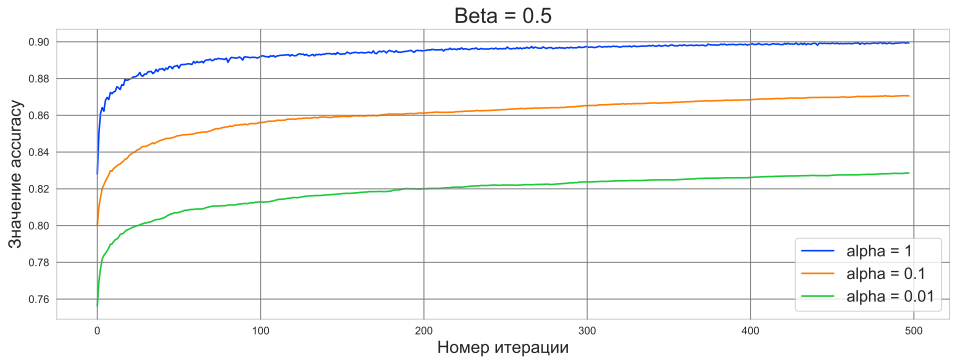
мя работы алгоритма уменьшилось. Если сравнивать конечные показатели, то для SGDClassifier без лемматизации значение accuracy составило 0.8916, а с лемматизацией точность повысилась до 0.8929. Время работы составило, соответственно, 40.78 и 30.14 секунд.

Пятый эксперимент

Теперь рассмотрим, как на качество алгоритма влияет применение Tfidf. Отметим, что размер признакового пространства не изменился. При этом скорость сходимости увеличилась: на то, чтобы сойтись, алгоритму потребовалось 25 секунд вместо 30.14 для алгоритма без применения TfIdf. Однако при этом пострадало качество: теперь точность алгоритма на валидации 0.8293. Получается, алгоритм сходится до того, как он достигает лучших показателей по качеству.

Теперь попробуем изменить параметры min-df и max-df и проверить, как это скажется на качестве алгоритма. Мы начинали с параметра 0.0001. Добавим в наше признаковое пространство больше объектов, уменьшив этот параметр до 0.00001. Снова применим TfidfTransformer. Теперь размер признакового пространства равен 89658. Время, затраченное на обучение, закономерно возросло. Также понизилось качество за счет учитывания редко встречающихся слов. Что будет, если выбрать промежуточное значение? Например, min-df = 0.00008. Тогда значение accuracy составит 0.8282. Снова качество ухудшилось. Давайте оставим этот параметр равным 0.0001. Проверим, улучшится ли результат, если изменить max-df. Чтобы остаться в



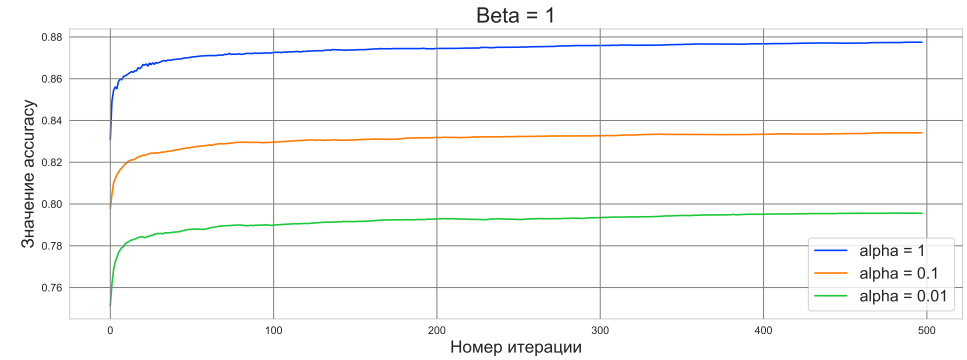


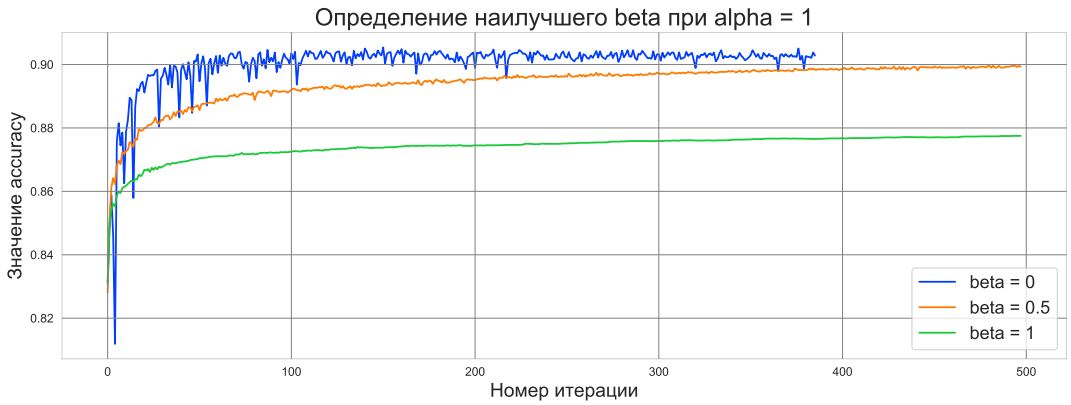
признаковом пространстве примерно той же размерности, выставим min-df = 0.00008, max-df = 0.1. Теперь признаковое пространство имеет размерность 18197 и обучается за 18.45 секунд, но качество понизилось до 0.828. При min-df = 0.0001, max-df = 0.1 качество составило 0.8287, что не сильно хуже, чем точность при максимальном max-df, а время сходимости сократилось до 18 секунд. Предлагаю остановиться на этих параметрах.

Теперь проверим, как на качество нашего алгоритма повлияет добавление n-грамм различных размерностей. Начнем с биграмм. Их добавление сразу расширило пространство признаков до 70671. Время обучения увеличилось до 34 секунд, качество упало до 0.8154. Что будет, если оставить только биграммы? Теперь размерность пространства 54676, а качество совсем упало до значения 0.79. Получается, отдельные слова все-таки играют большую роль для предсказания, чем биграммы. На примере биграмм мы уже заметили, что их добавление ухудшает предсказание. Все-таки попробуем добавить и 3-граммы, чтобы проанализировать, насколько увеличится признаковое пространство и время работы алгоритма. Время обучения увеличилось до 55.65 секунд, качество составило 0.8136, а размер пространства 105225. Получается, добавление n-грамм увеличивает время обучения примерно в 1.5^n раз.

Финальный эксперимент

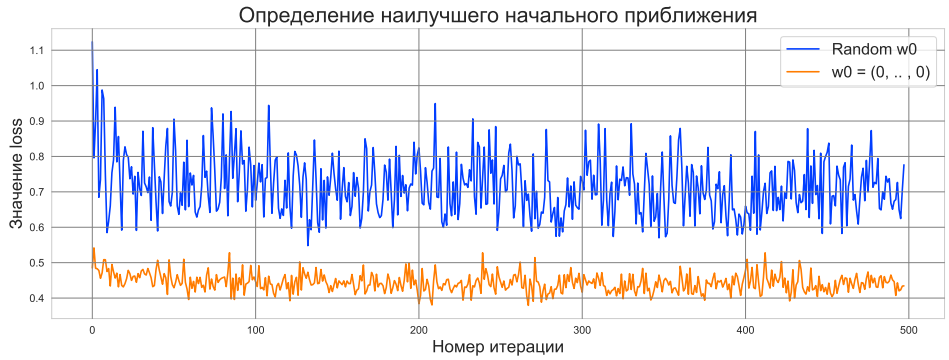
Рассмотрим финальную модель: для предобработки будем использовать лемматизацию с исключением стоп-слов. SGDClassifier с параметрами $\alpha = 1, \beta =$

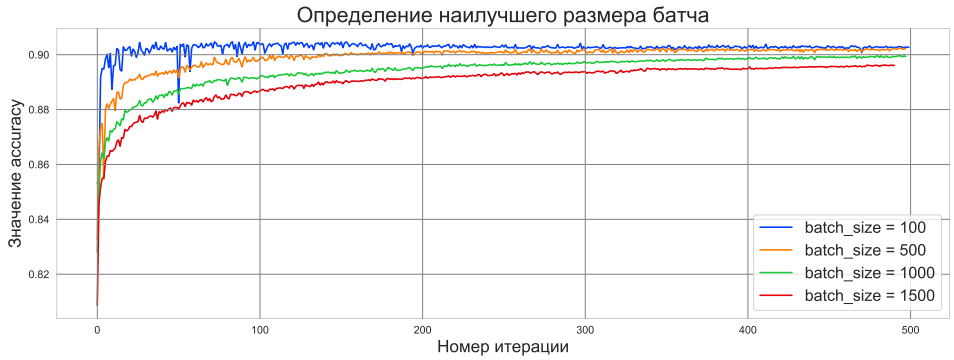


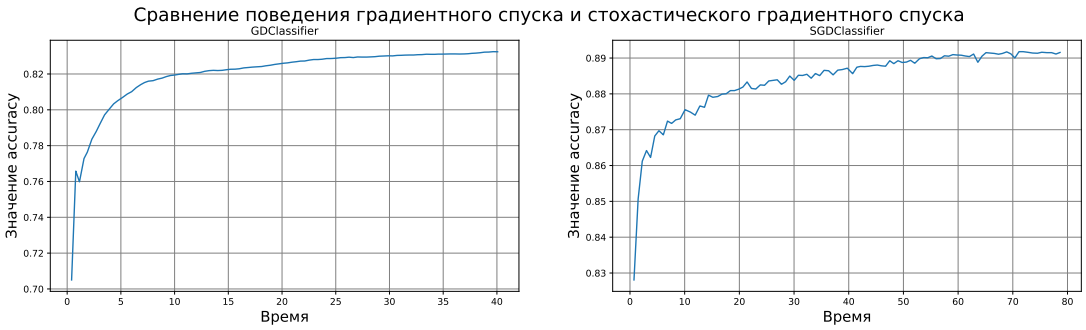


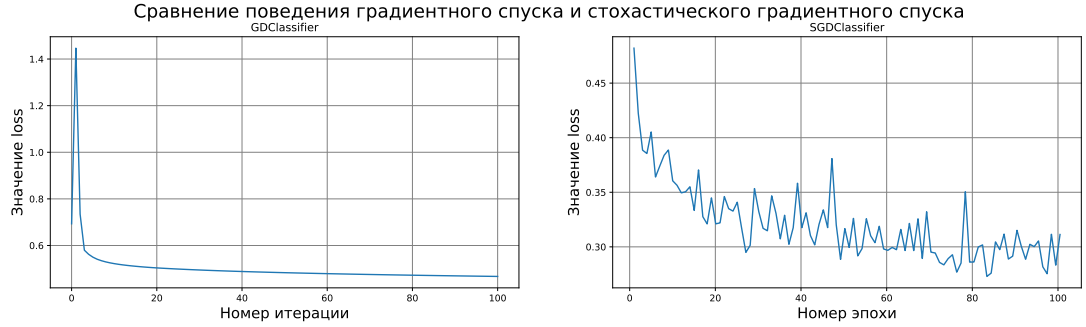
$0.5, batch_size = 1000$. Максимальное количество итераций примем равным 1000. Заметим, что до этого мы принимали везде параметр регуляризации равным 0. Попробуем подобрать его сейчас: с нулевым значением параметра получаем accuracy = 0.87. При увеличении коэффициента регуляризации до 1 точность падает до 0.81. Если попытаться уменьшать коэффициент, точность растет, пока при значении 0.0001 не превышает показатель accuracy для нулевой регуляризации. Получаем лучшее предсказание 0.872.

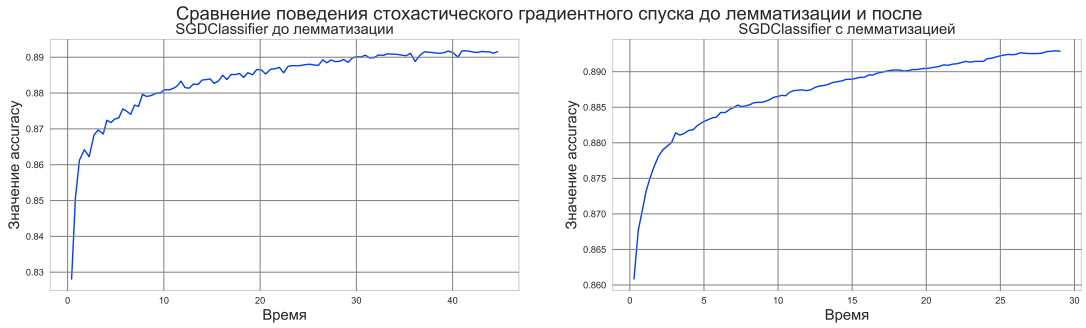
Проанализируем, на каких объектах алгоритм ошибается. Рассмотрим случаи, когда модель ошибочно относит комментарии к ошибочным. Я заметила, что в некоторых случаях, кажется, присутствуют ошибки разметки датасета. Так, например, комментарий "i will burn you to hell if you revoke my talk page access"считается нетоксичным, что на мой взгляд неверно. Также модель относит к токсичным слова с опечатками и случайные наборы букв. Некоторые ошибки были допущены из-за того, что в предложении использовались словаи словосочетания с негативной окраской, хотя фраза целиком не была токсичной: lazy, 'are not a race'. Теперь проанализируем случаи, когда алгоритм ошибочно считает комментарии нетоксичными. Это происходит тогда, когда токсичными являются очень редко встречающиеся слова, либо когда токсичный комментарий передан в "вежливой"форме. В целом, можно сделать вывод, что модель неспособна отследить тонкие случаи.

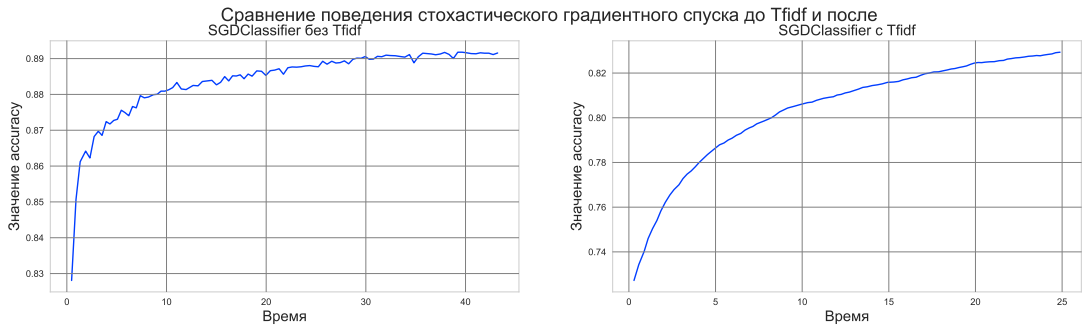

Определение наилучшего начального приближения
Random w0
w0 = (0, .. , 0)
Значение loss
Номер итерации


Определение наилучшего размера батча
batch_size = 100
batch_size = 500
batch_size = 1000
batch_size = 1500
Значение accuracy
Номер итерации


Сравнение поведения градиентного спуска и стохастического градиентного спуска
GDClassifier
SGDClassifier
Значение accuracy
Время


Сравнение поведения градиентного спуска и стохастического градиентного спуска
GDClassifier
SGDClassifier
Значение loss
Номер итерации
Номер эпохи


Сравнение поведения стохастического градиентного спуска до лемматизации и после
SGDClassifier до лемматизации
SGDClassifier с лемматизацией
Значение accuracy
Время


Сравнение поведения стохастического градиентного спуска до Tfidf и после
SGDClassifier без Tfidf
SGDClassifier с Tfidf
Значение accuracy
Время