Задание 2. Градиентные методы обучения линейных моделей. Применение линейных моделей для определения токсичности комментария.

Янаков Дмитрий, 317 группа

9 ноября, 2021

1 Введение

В этом задании будут изучены градиентные методы обучения линейных моделей: «градиентный спуск» и «стохастический градиентный спуск», а именно: проведены эксперименты на датасете, содержащим комментарии из раздела обсуждений английской Википедии. Для начала будут выбраны оптимальные параметры для минимизации функции потерь и максимизации точности классификации. Затем, проведя дополнительное исследование, используя лемматизацию и другое векторное представление для комментариев, будет подобрана оптимальная стратегия для классификации. И под конец мы рассмотрим наиболее частые ошибки алгоритма и укажем их общие черты.

2 Пояснения к заданию

Приведём некоторые теоретические выкладки.

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

   Тогда функция потерь имеет следующий вид:

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

   Выведем формулу градиента функции потерь:

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

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

2. Теперь рассмотрим задачу мультиномиальной логистической регрессии, т.е. $\mathbb{Y} = \{1, ..., K\}$.

   В этом случае функция потерь будет имеет следующий вид:

$$Q(X, w) = -\frac{1}{l} \sum_{i=1}^{l} \ln \frac{\exp(\langle w_{y_i}, x_i \rangle)}{\sum_{k=1}^{K} \exp(\langle w_k, x_i \rangle)} = \frac{1}{l} \sum_{i=1}^{l} \left[\ln \sum_{k=1}^{K} \exp(\langle w_k, x_i \rangle) - \langle w_{y_i}, x_i \rangle \right]$$

   Выведем формулу градиента функции потерь:

$$dQ_{w_j}(X, w) = \frac{1}{l} \sum_{i=1}^{l} \left[\frac{\sum_{k=1}^{K} \exp(\langle w_k, x_i \rangle) d \langle w_k, x_i \rangle}{\sum_{k=1}^{K} \exp(\langle w_k, x_i \rangle)} - d \langle w_{y_i}, x_i \rangle \right]$$

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

$$\Rightarrow \nabla_{w_j} Q(x, w) = \frac{1}{l} \sum_{i=1}^{l} \left[\frac{\exp(\langle w_j, x_i \rangle) x_i}{\sum_{k=1}^{K} \exp(\langle w_k, x_i \rangle)} - x_i \left[y_i = j \right] \right]$$

3. Покажем, что при количестве классов = 2, задача мультиномиальной логистической регрессии сводится к бинарной логистической регрессии.

   Действительно, для бинарного случая, имеем следующую вероятность принадлежности объекта x классу 1:

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

   В случае двух классов $\mathbb{Y} = \{1, -1\}$, для мультиномиальной регрессии аналогичную вероятность можно выразить следующим образом:

   $$\mathbb{P}(y = 1|x) = \frac{\exp(\langle w_1, x \rangle)}{\exp(\langle w_1, x \rangle) + \exp(\langle w_{-1}, x \rangle)} = \frac{1}{1 + \exp(\langle w_{-1} - w_1, x \rangle)}$$

   Получаем, что задача мультиномиальной регрессии переходит в задачу бинарной логистической регресии с вектором весов $w = -(w_{-1} - w_1) = w_1 - w_{-1}$.

3 Эксперименты

Перед тем, как проводить эксперименты, произведём некоторую обработку текста: приведём все комментарии к нижнему регистру и заменим в них все символы, не являющиеся буквами и цифрами, на пробелы.

Сами комментарии будут представлены векторно с помощью модели **BagOfWords** (с параметром **min_df** $= 10^{-4}$). В дальнейшем будет приведено сравнение с моделью **TF-IDF**.

По умолчанию, будем использовать следующие параметры для градиентных методов обучения: **максимальное количество итераций** $= 100$, **точность, по достижении которой, необходимо прекратить оптимизацию** $= 10^{-5}$, **частоту обновления (для стохастического градиентного спуска)** $= 0.3$, а также **коэффициент регуляризации** $= 0.01$.

Отметим также, что подбор параметров будет проводиться по анализу следующих зависимостей:

- зависимость значения функции потерь от реального времени работы метода
- зависимость значения функции потерь от итерации метода (эпохи в случае стохастического варианта)
- зависимость точности (accuracy) от реального времени работы метода
- зависимость точности (accuracy) итерации метода (эпохи в случае стохастического варианта)

3.1 Эксперимент 1

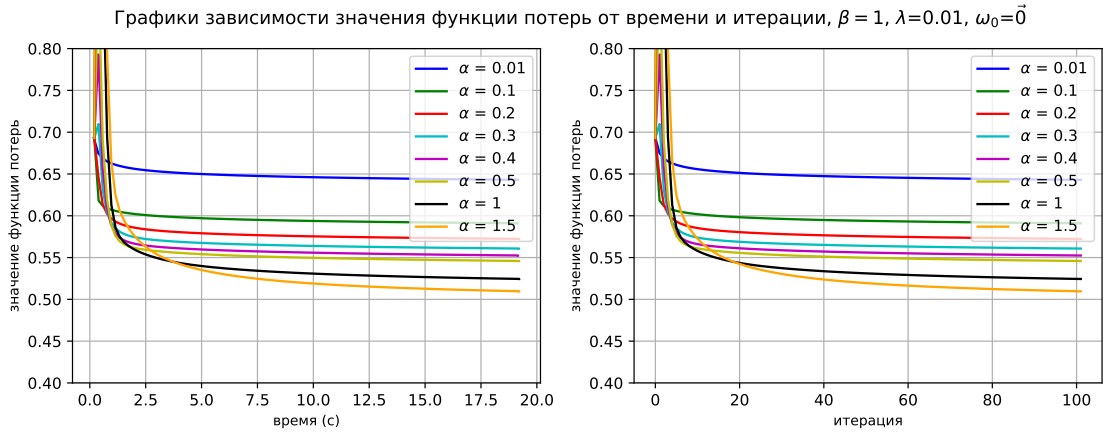
Рассмотрим следующие параметры для градиентного спуска и подберем оптимальные:

- параметр размера шага α
- параметр размера шага β
- начальное приближение w_0

где α и β взяты из формулы для параметра темпа обучения $\eta_k = \frac{\alpha}{k^\beta}$.

3.1.1 Параметр размера шага α

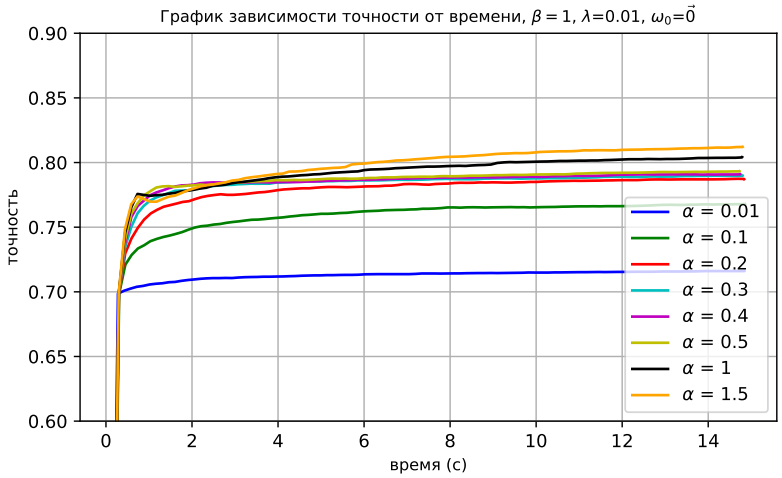
Зафиксируем $\beta = 1$, $w_0 = \vec{0}$ и рассмотрим результаты работы алгоритма при разных α:



Сразу отметим тот факт, что графики зависимости от времени и итераций очень похожи, поэтому в дальнейшем мы будем опускать графики зависимости от итераций. Они приведены в **Приложении А**.

Из графиков видно: чем больше α, тем быстрее минимизируется функция потерь. Также заметим, что $\alpha < 0$ не рассматривается, поскольку параметр темпа обучения должен быть положительным.

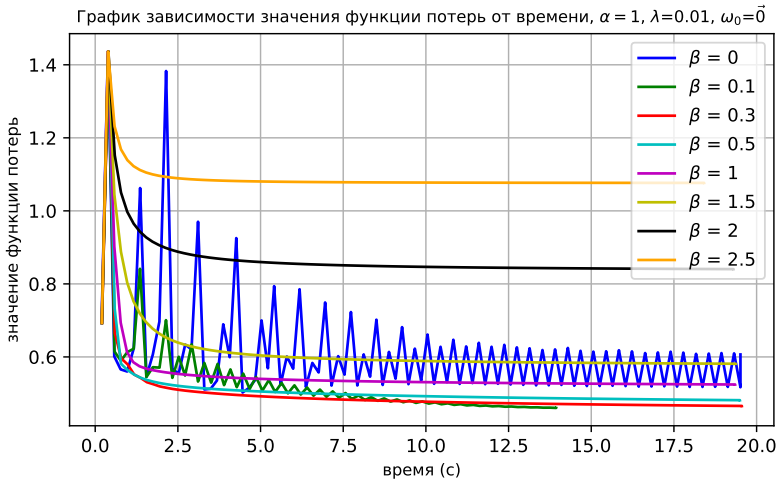
Для исследования точности (accuracy) разобьем обучающую выборку на две подвыборки в отношении 7:3. Это разбиение будет использовано и далее для подбора других параметров.



Ситуация получается аналогичная: чем больше α, тем быстрее и больше достигается точность.

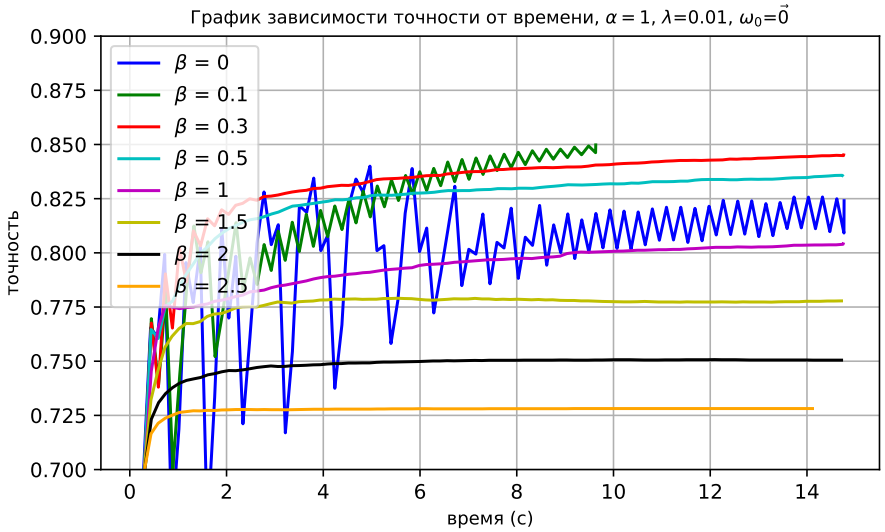
3.1.2 Параметр размера шага β

Зафиксируем $\alpha = 1$, $w_0 = \vec{0}$ и рассмотрим результаты работы алгоритма при разных β:



Из графика видно: чем меньше β, тем быстрее минимизируется функция потерь, однако при $\beta = 0$ видны осцилляции (на самом деле они есть и при отрицательных β), которые связаны с тем, что алгоритму не хватает количества итераций для того, чтобы график вышел на плато. Также можно заметить, что кривая при $\beta = 0.1$ обрывается раньше других. Это связано с тем, что точность, по достижении которой, необходимо прекратить оптимизацию, была достигнута.

Для точности на валидационной выборке получаются следующие результаты:



Выводы аналогичны предыдущему графику: при меньших β достигается большая точность, однако при совсем малых β график не успевает выйти на плато.

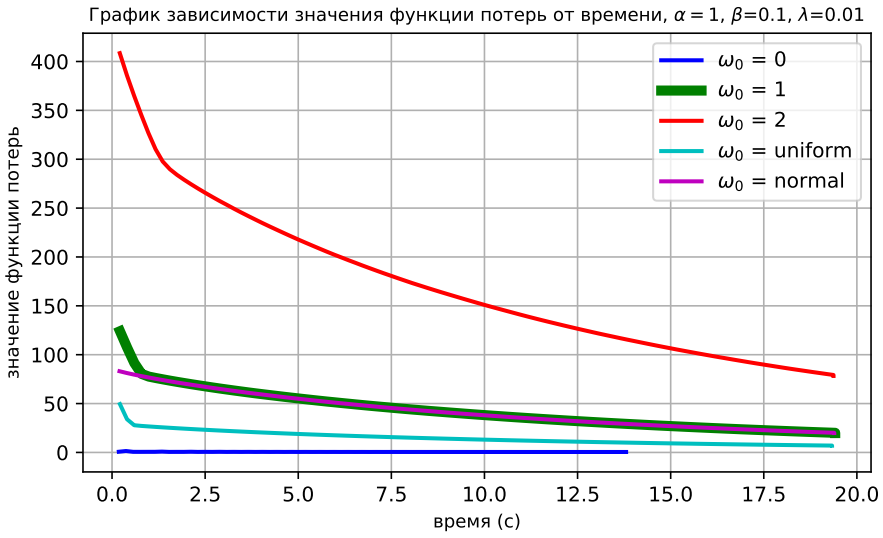
3.1.3 Параметр w_0

Зафиксируем $\alpha = 1$, $\beta = 0.1$ и рассмотрим результаты работы алгоритма при разных начальных приближениях w_0.

Возьмём следующие значения для w_0:

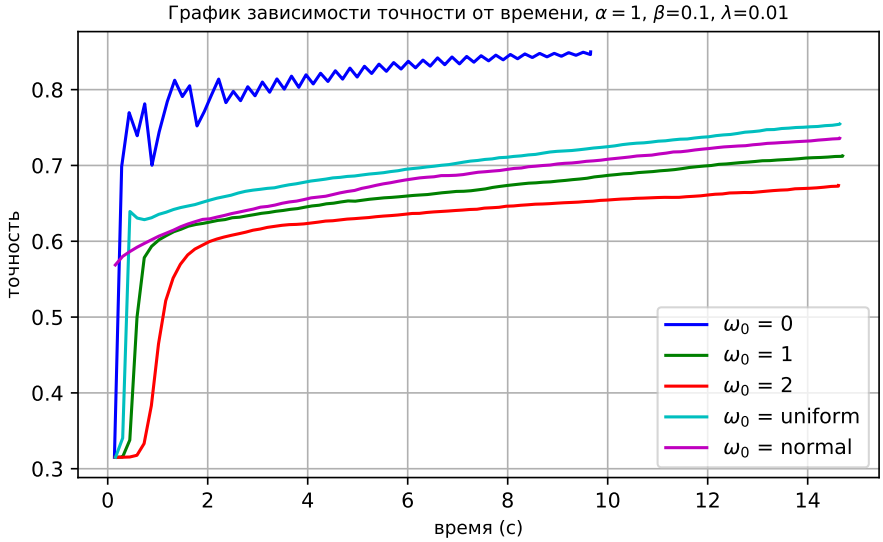
- $w_0 = (0, ..., 0)$. Обозначение: $w_0 = 0$.
- $w_0 = (1, ..., 1)$. Обозначение: $w_0 = 1$.
- $w_0 = (2, ..., 2)$. Обозначение: $w_0 = 2$.
- $w_0 \sim U(0, 1)$. Обозначение: $w_0 = \text{uniform}$.
- $w_0 \sim N(0, 1)$. Обозначение: $w_0 = \text{normal}$.

Получим следующий результат:



Видно, что при увеличении нормы начального приближения, в первое время функция потерь принимает большие значения. Оптимальным является нулевое начальное приближение, так как функция потерь практически сразу выходит на плато и принимает меньшие значения, по сравнению с другими. Также при нём быстрее достигается оптимизирующая точность. Можно заметить, что функция потерь с начальным приближением, распределенным нормально, ведёт себя примерно так же, как функция потерь с начальным приближением $w_0 = 1$.

Для точности получаются следующие графики:



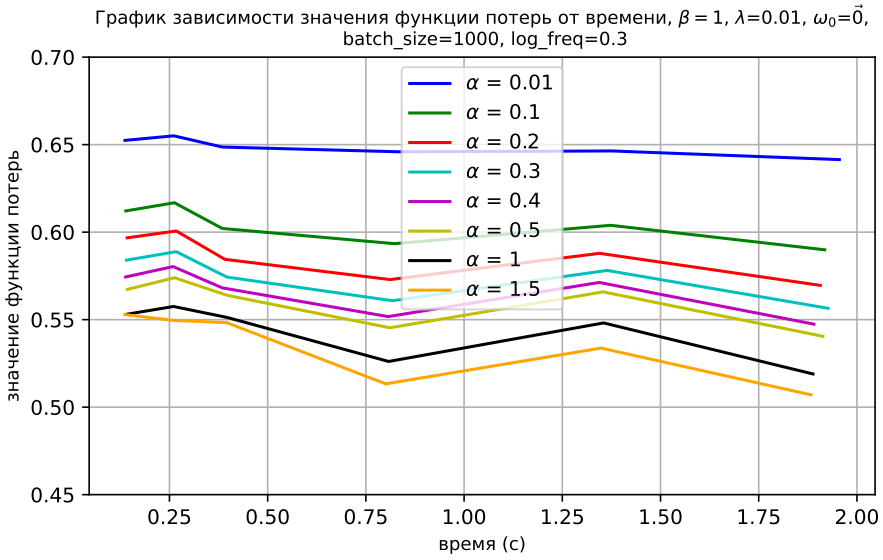
В данном случае хоть и видны осцилляции для нулевого начального приближения, график достаточно быстро выходит на плато и показывает результаты лучше, по сравнению с другими.

3.2 Эксперимент 2

В эксперименте 1 мы рассматривали параметры для градиентного спуска. Для стохастического градиентого спуска рассматриваются те же самые параметры, но добавляется еще один — размер подвыборки **batch_size**.

3.2.1 Параметр размера шага α

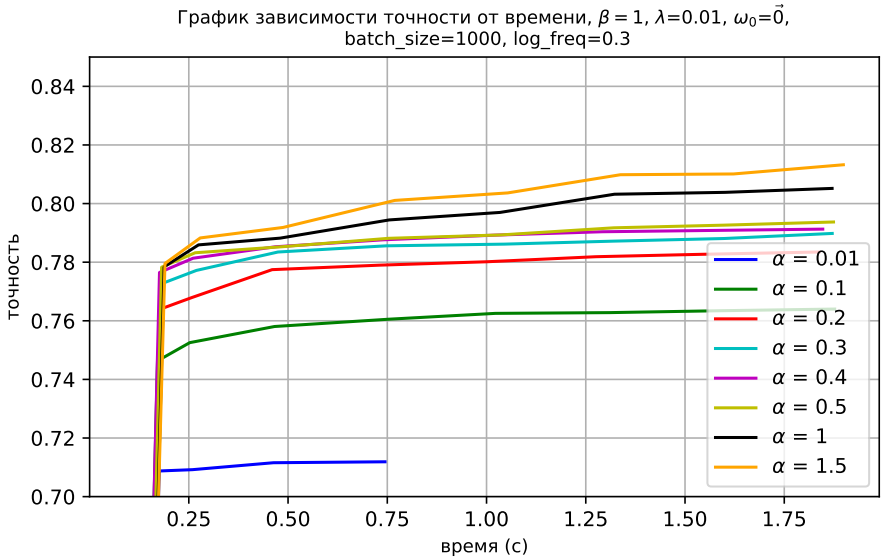
Зафиксируем $\beta = 1$, $w_0 = \vec{0}$, batch_size $= 1000$ и рассмотрим результаты работы алгоритма при разных α:



Зависимость значений функций потерь от времени аналогична градиентному случаю: при больших α функция потерь принимает меньшие значения.

Также, графики зависимости от эпох похожи на графики зависимости от времени, поэтому они приведены в **Приложении В**.

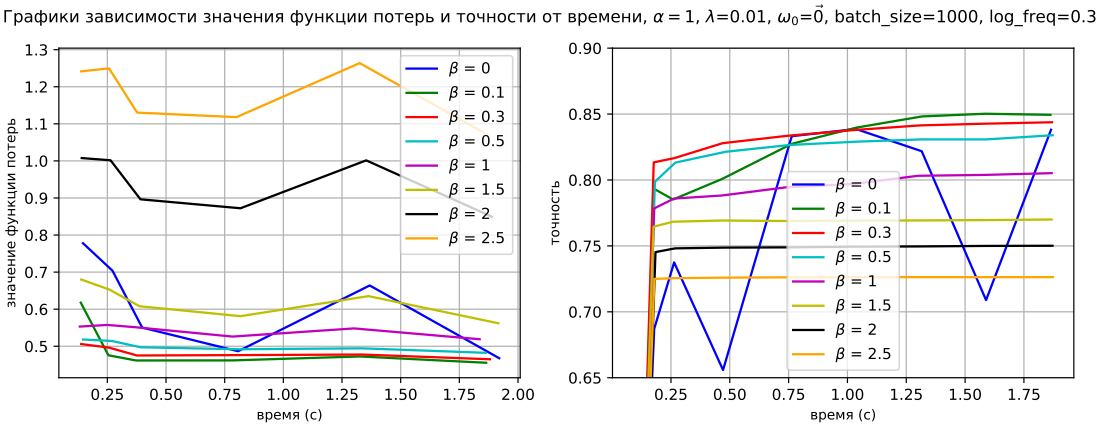
Для точности получается следующий результат:



Зависимость полностью аналогична предыдущему случаю, но что интересно: при $\alpha = 0.01$ алгоритм проработал меньше всех. Это значит, что в стохастическом случае быстрее была достигнута оптимизирующая точность, но, как оказалось, не точность на валидационной выборке.

3.2.2 Параметр размера шага β

Зафиксируем $\alpha = 1$, $w_0 = \vec{0}$, batch_size $= 1000$ и рассмотрим результаты работы алгоритма при разных β:

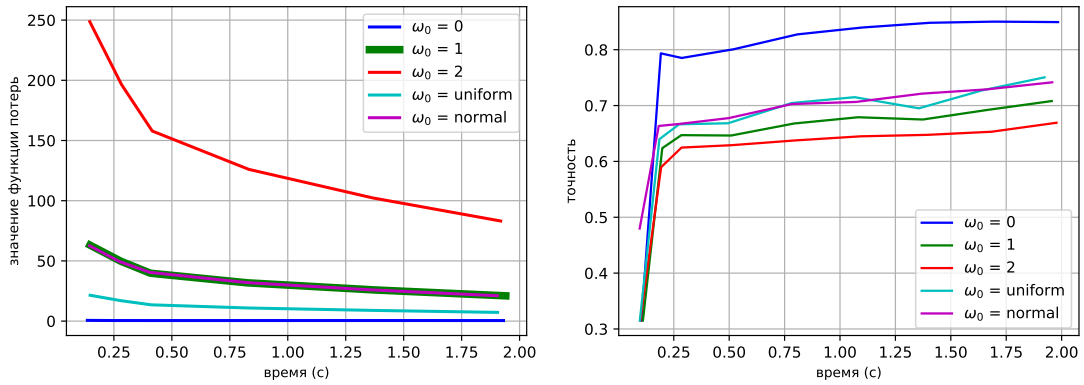


Здесь уже не наблюдаются сильные осцилляции (однако все равно присутствуют) при $\beta = 0$, так как на это еще влияют параметры **batch_size** и **log_freq** (частота обновления), однако ситуация схожа с градиентным спуском: малые β показывают лучший результат. Точность на валидационной выборке приблизительно равна **85**%.

3.2.3 Параметр w_0

Зафиксируем $\alpha = 1$, $\beta = 0.1$, batch_size $= 1000$ и рассмотрим результаты работы алгоритма при тех же самых w_0, как и в эксперименте 1:

Графики зависимости значения функции потерь и точности от времени, $\alpha = 1$, λ=0.01, ω_0=$\vec{0}$, batch_size=1000, log_freq=0.3

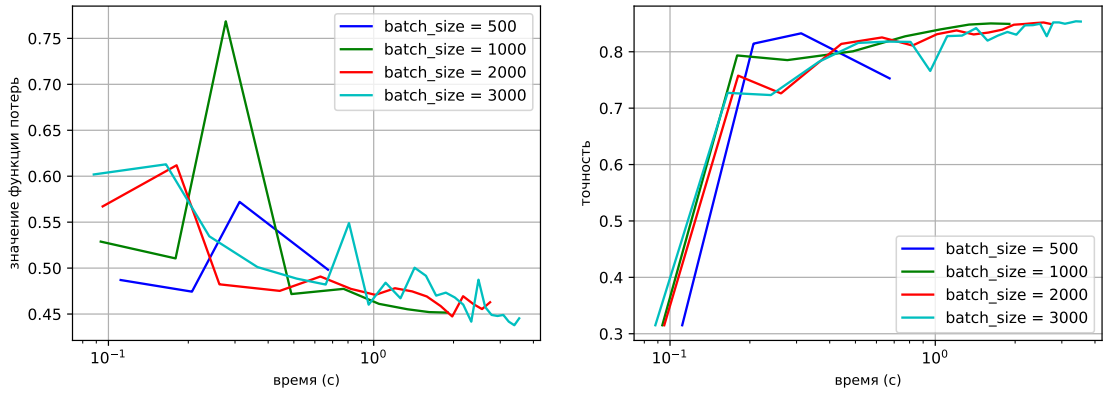


Нулевое начальное приближение показывает наилучший результат, как для минимизации функции потерь, так и для максимизации точности на валидационной выборке. Функции потерь для $w_0 = 1$ и $w_0 = normal$ ведут себя даже более похоже, чем в случае градиентного спуска.

3.2.4 Параметр batch_size

Зафиксируем $\alpha = 1$, $\beta = 0.1$, $w_0 = \vec{0}$, и рассмотрим результаты работы алгоритма при разных batch_size:

Графики зависимости значения функции потерь и точности от времени, $\alpha = 1$, $\beta = 0.1$, λ=0.01, ω_0=$\vec{0}$, log_freq=0.3



Видно, что при малых размерах подвыборки, алгоритм быстро прекращает работу, не добившись хорошего результата. С увеличением размера минимизируется функция потерь и увеличивается точность. Однако тут не прослеживается хорошей зависимости от размера, поскольку в разные моменты времени, алгоритм с меньшим размером подвыборки может выдавать результат лучше, чем алгоритм с большим размером, и как видно, функция потерь не сразу выходит на плато.

3.3 Эксперимент 3

Теперь, когда мы подробно изучили зависимость функции потерь и точности от различных параметров для градиентного спуска (GD) и стохастического градиентного спуска (SGD), мы можем сравнить их поведение.

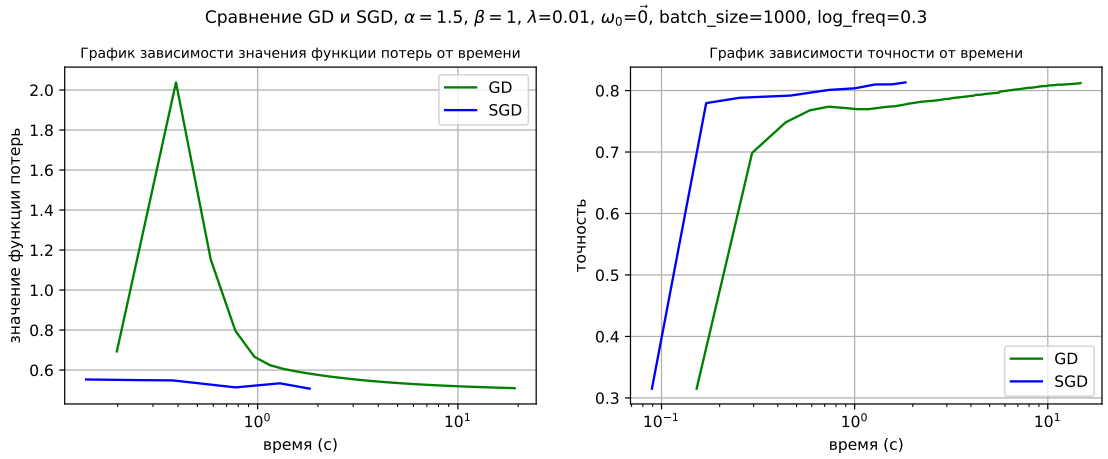
Зафиксируем следующие параметры: $\alpha = 1.5$, $\beta = 1$, $w_0 = \vec{0}$, batch_size = 1000.

Получим следующие результаты:

	Время работы, с	Точность на валидационной выборке, %	Значение функции потерь
GD	14.80	81.20	0.50
SGD	1.89	81.32	0.50

Время работы стохастического градиентного спуска гораздо меньше времени работы обычного градиентного спуска. Это достигается за счёт того, что на каждой итерации алгоритма выбираюся не все объекты, а только некоторое их подмножество. Точность у SGD чуть выше, но итоговое значение функций потерь одинаково.

Для наглядности, проиллюстрируем результат в виде графиков:



3.4 Эксперимент 4

Попробуем применить алгоритм лемматизации (приведение слов в начальную форму) с удалением стоп-слов и исследуем, как такая предобработка данных повлияет на точность классификации, время работы алгоритма и размерность признакового пространства.

На примере покажем, как изменяется предложение:

- до лемматизации и удаления стоп-слов: *how could i post before the block expires the funny thing is you think i m being uncivil*
- после лемматизации и удаления стоп-слов: *could post block expires funny thing think uncivil*

Как видно, размер предложения существенно уменьшился, что вероятно скажется на размерности признакового пространства. Действительно: до лемматизации и удаления стоп-слов, размерность признакового пространства составляла — **16050**, после — **14291**.

Сравним время и точность для разных алгоритмов с теми же параметрами, как в эксперименте 3:

	Время работы, с	Точность классификации, %
GD без предобработки	14.81	81.20
GD с предобработкой	10.12	82.01
SGD без предобработки	1.90	81.32
SGD с предобработкой	1.72	81.45

Как видно из результатов, лемматизация и удаление стоп-слов действительно помогли увеличить точность классификации и уменьшить время работы как в случае градиентного спуска, так и в случае стохастического градиентного спуска. Причём в случае градиентного спуска изменения более значительные.

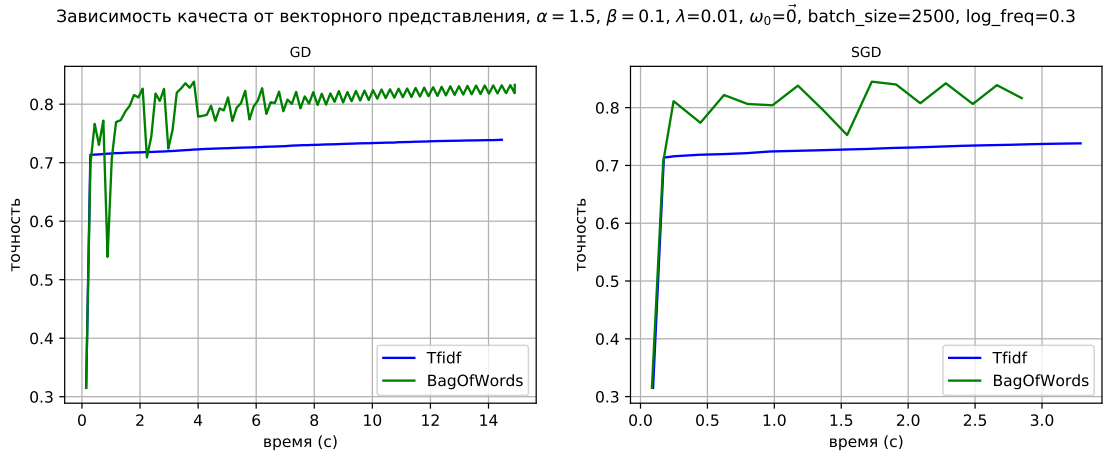
Связано это, конечно, с тем, что размерность признакового пространства уменьшилась и алгоритм обрабатывает меньший объем данных. К тому же, после обработки исчезли менее значительные слова, что также повлияло на точность.

3.5 Эксперимент 5

В начале мы упомянули, что будем использовать векторное представление для комментариев с помощью модели **BagOfWords** (**BOF**). Попробуем теперь использовать модель **TF-IDF** и сравнить качество, время работы алгоритмов и размерность признакового пространства.

Поскольку словарь при новом представлении не изменился, то не изменилась и размерность признакового пространства. Она также равна **16050**.

Рассмотрим, как меняется точность и время работы алгоритмов в зависимости от модели при тех же параметрах, как в эксперименте 3, кроме параметра batch_size. В этом эксперименте batch_size = 2500.



Видно, что точность на модели **BOF**, больше чем на **TF-IDF** как для градиентного спуска, так и для стохастического градиентного спуска. Однако время работы примерно одинаковое. В случае GD **TF-IDF** работает быстрее, чем **BOF**; в случае SGD **TF-IDF** работает медленнее.

Важно отметить, что данный результат справедлив только для batch_size = 2500. Если варьировать данный параметр, то будет меняться продолжительность работы алгоритмов, но точность **BOF** будет оставаться больше.

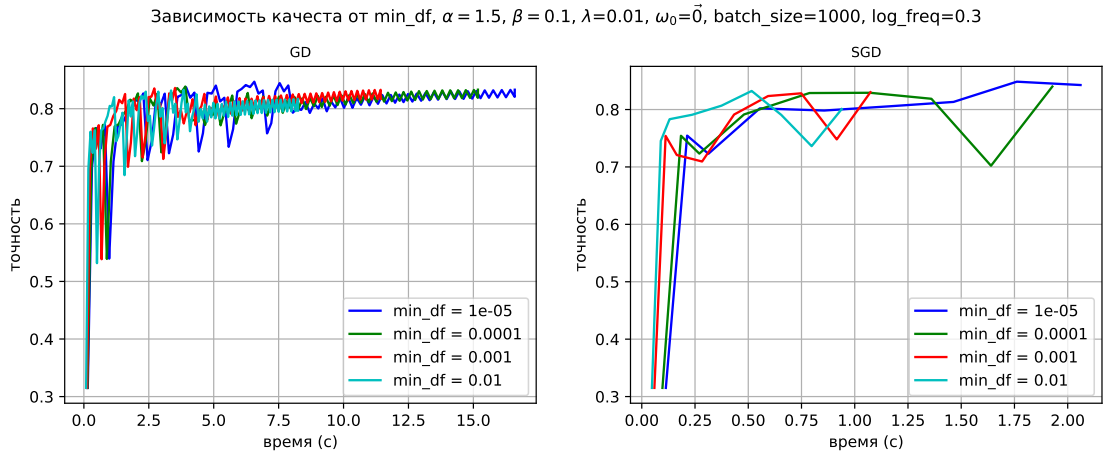
3.6 Эксперимент 6

Исследуем зависимость качества, времени работы и размерности признакового пространства в зависимости от следующих параметров модели **BOF**:

- min_df
- max_df

3.6.1 Параметр min_df

Проиллюстрируем зависимость точности от времени для различных значений параметра:

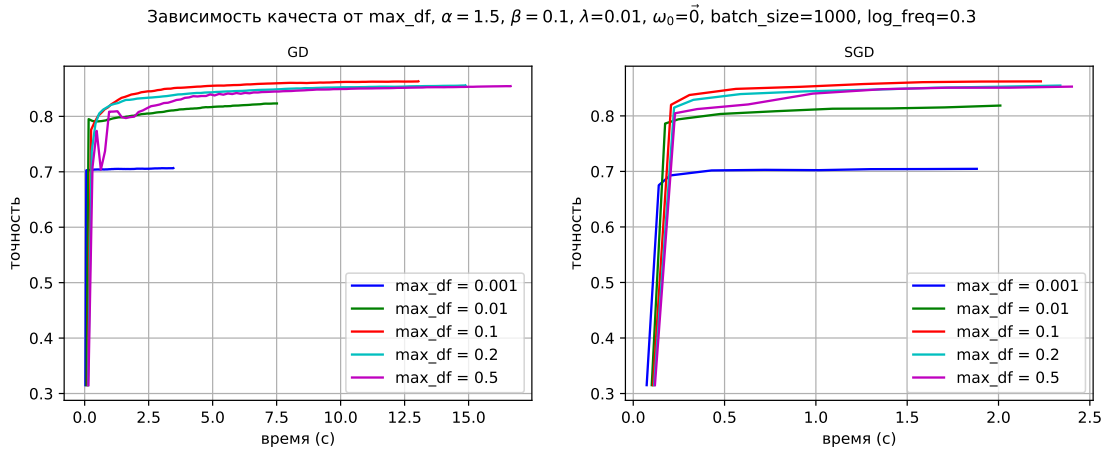


Для случая GD видно, что при увеличении значении параметра, уменьшается точность и итоговое время работы алгоритма. Связано это конечно с тем, что размерность признакового пространства падает, поскольку исчезают слова с частотой меньше, чем заданный параметр.

Для случая SGD график с наименьшим значением параметра быстрее всех выходит на плато. По времени работы ситуация аналогична с GD.

3.6.2 Параметр max_df

Проиллюстрируем зависимость точности от времени для различных значений параметра:



В случае слишком малого значения параметра высокой точности не наблюдается. При увеличении значения параметра, точность растёт, но увеличивается и время работы алгоритма, хотя размерность признакового пространства, как будет показано далее, увеличивается ненамного.

Поскольку вероятность того, что в комментариях будут одни токсичные слова, конечно, же мала, то увеличение точности связано вероятнее всего с тем, что удаляются слова с частотой большей, чем заданный параметр.

3.6.3 Сравнение

Сравним значения для размерностей признаковых пространств при разных значениях параметров:

	min_df				max_df				
	0.1	0.01	0.0001	0.00001	0.001	0.01	0.1	0.2	0.5
Размерность	89658	16050	3736	568	85922	89090	89603	89638	89655

Заметим, что размерность уменьшается гораздо быстрее в случае варьирования параметра min_df, то есть варьирования наименьшей частоты, при которой слово добавляется в словарь.

3.7 Эксперимент 7

После исследования многих параметров и моделей вернёмся к тестовой (не валидационной) выборке и выберем лучший алгоритм для неё.

Для достижения более правдоподобной точности, увеличим максимальное количество итераций до 1000.

Используем стохастический градиентный спуск с моделью **BagOfWords** (и параметром min_df = 0.001) и следующими параметрами: $\alpha = 1.5$, $\beta = 0.1$, $w_0 = \vec{0}$ и batch_size = 3000.

Точность на тестовой выборке составила — **82.93%**, а итоговое время работы — **68.59с**.

Рассмотрим некоторые типичные ошибки, которые допускает алгоритм:

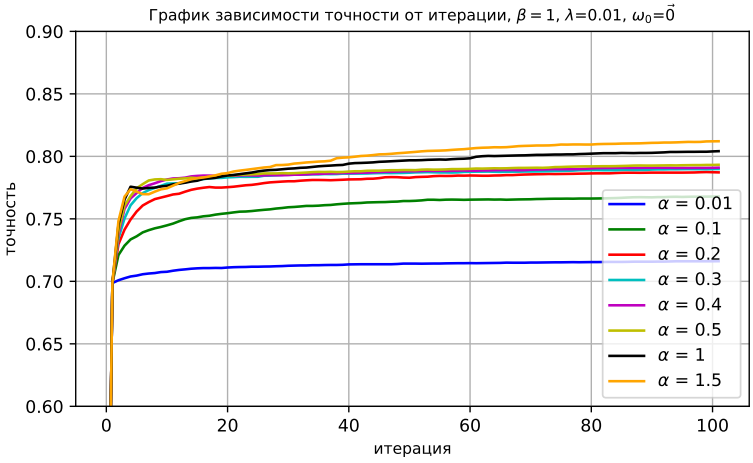
- Например, комментарий «*Dear god this site is horrible.*» был классифцирован, как токсичный, хотя таковым не является. Скорее всего, это из-за того, что он содержит слово «*horrible*».
- Другой пример: комментарий «*That's helpful. MOS be damned, Thecodingproject thinks it's '10x' worse.*» был классифцирован, как нетоксичный, хотя является токсичным. Возможно, алгоритм не считает слово «*damn*» весомым, для того, чтобы отнести комментарий к токсичному. И это резонно, поскольку «*damn*» может выражать так же и удивление.

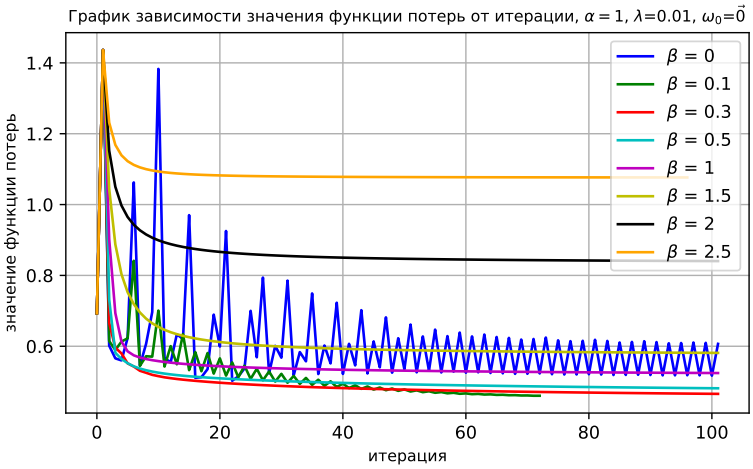
То есть, в основном, ошибки допускаются из-за того, что алгоритм попросту не понимает смысла предложения. У него есть слова, которые прибавляют веса к токсичности, а есть те, которые уменьшают. Также есть случаи неанглийских комментариев, на которых алгоритм, очевидно, может легко допустить ошибку, поскольку обучается он преимущественно на английских комментариях.

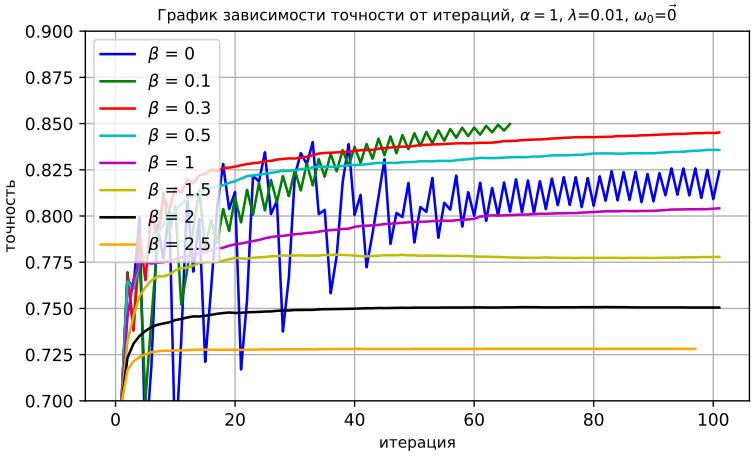
4 Выводы

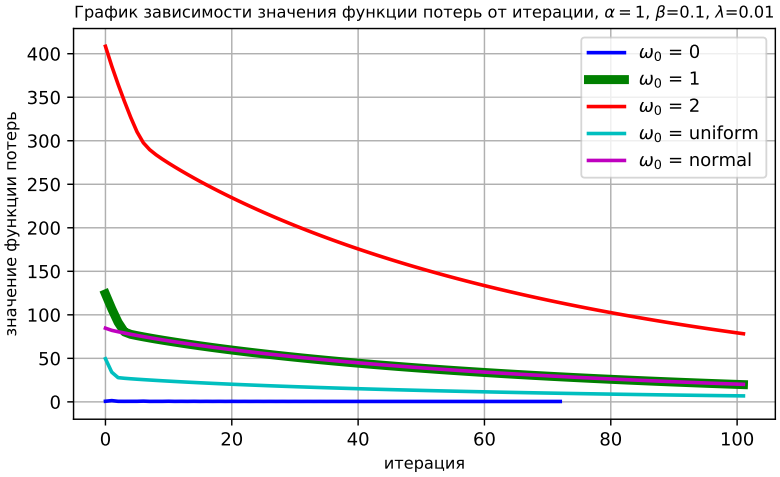
Итак, мы рассмотрели датасет комментариев из раздела обсуждений английской Википедии и провели эксперименты над ним для выявления наилучшего алгоритма для классификации. Изначально мы определили оптимальные параметры как для градиентного спуска, так и для стохастического градиентного спуска. Потом попробовали изменить векторное представление для комментариев, а также применили лемматизацию для повышения точности классификации. Важно отметить, что, точность, которую мы получили — скорее всего далеко не предел, поскольку в наших экспериментах мы фиксировали какие-то параметры, а какие-то варьировали. Для достижения большей точности нужно рассматривать комбинации различных значений параметров. В заключение были отмечены некоторые ошибки, которые свойственны нашему алгоритму.

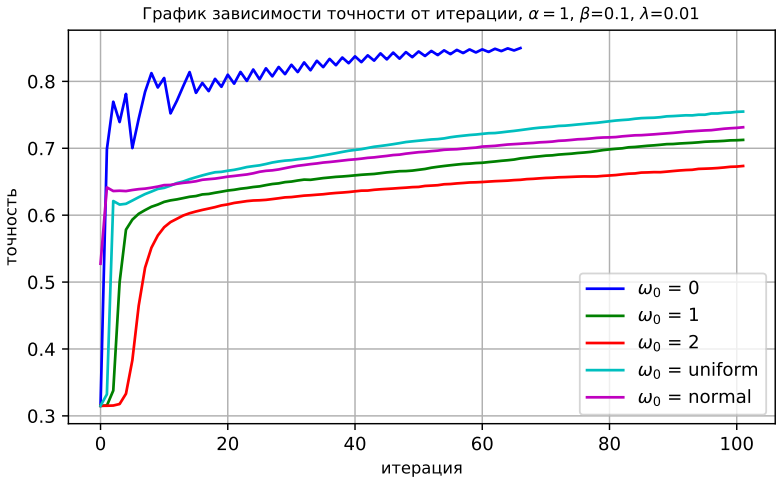
5 Приложение А. Градиентный спуск.



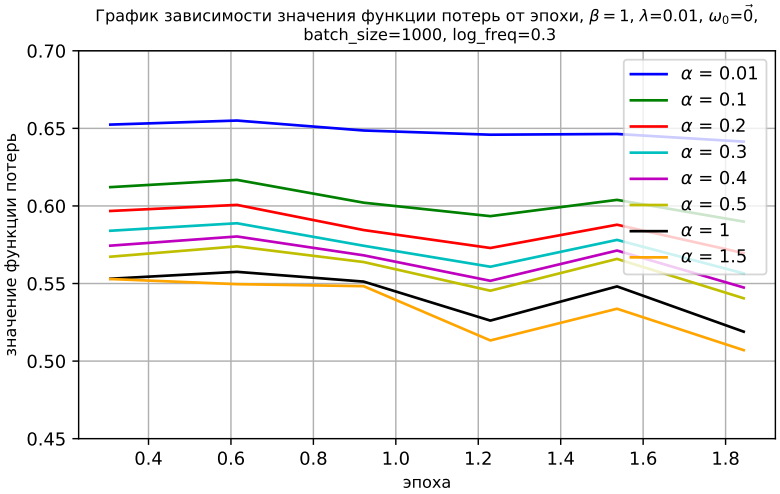








6 Приложение B. Стохастический градиентный спуск.



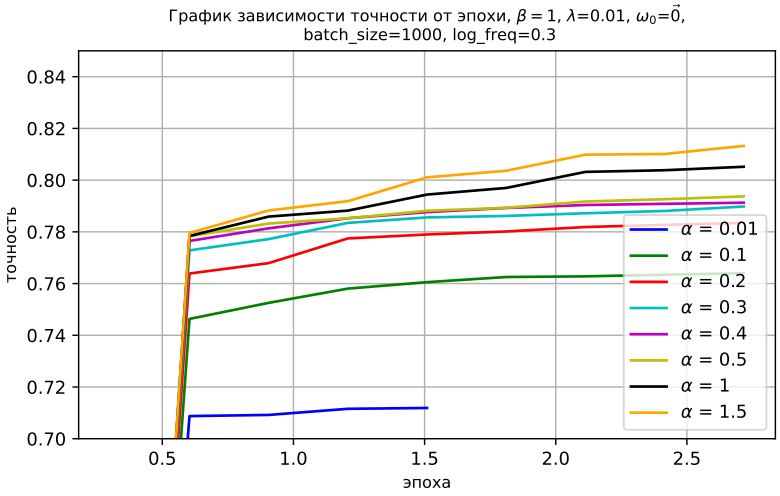


График зависимости точности от эпохи, $\beta = 1$, λ=0.01, ω_0=$\vec{0}$,
batch_size=1000, log_freq=0.3
точность
эпоха
α = 0.01
α = 0.1
α = 0.2
α = 0.3
α = 0.4
α = 0.5
α = 1
α = 1.5

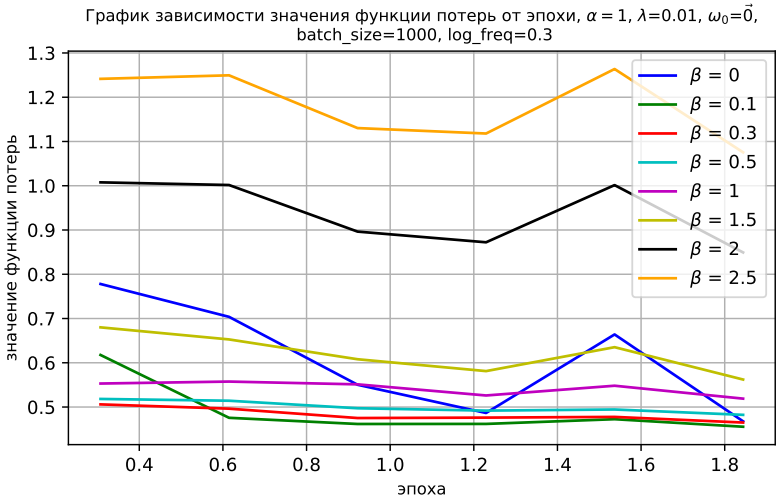


График зависимости значения функции потерь от эпохи, $\alpha = 1$, λ=0.01, ω_0=$\vec{0}$,
batch_size=1000, log_freq=0.3
значение функции потерь
эпоха
β = 0
β = 0.1
β = 0.3
β = 0.5
β = 1
β = 1.5
β = 2
β = 2.5

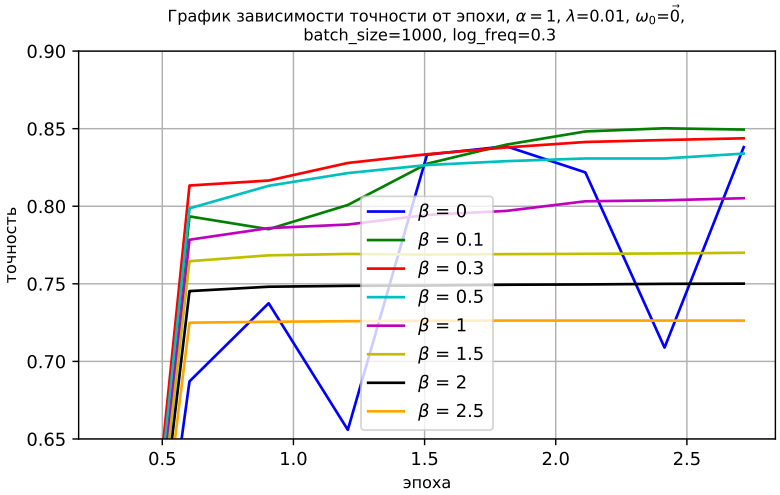


График зависимости точности от эпохи, $\alpha = 1$, λ=0.01, ω_0=$\vec{0}$,
batch_size=1000, log_freq=0.3
точность
эпоха
β = 0
β = 0.1
β = 0.3
β = 0.5
β = 1
β = 1.5
β = 2
β = 2.5

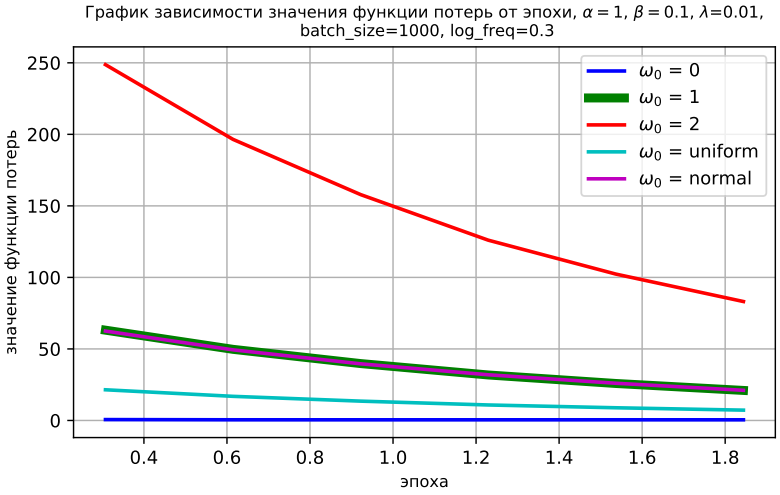


График зависимости значения функции потерь от эпохи, $\alpha = 1$, $\beta = 0.1$, λ=0.01,
batch_size=1000, log_freq=0.3
значение функции потерь
эпоха
ω_0 = 0
ω_0 = 1
ω_0 = 2
ω_0 = uniform
ω_0 = normal

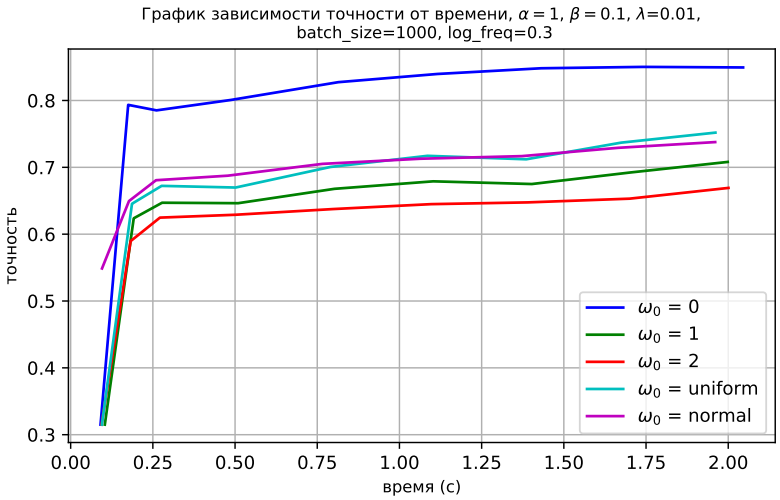


График зависимости точности от времени, $\alpha = 1$, $\beta = 0.1$, λ=0.01,
batch_size=1000, log_freq=0.3
точность
время (с)
ω_0 = 0
ω_0 = 1
ω_0 = 2
ω_0 = uniform
ω_0 = normal


График зависимости значения функции потерь от эпохи, $\alpha = 1$, $\beta = 0.1$, λ=0.01,
ω_0=$\vec{0}$, log_freq=0.3
значение функции потерь
эпоха
batch_size = 500
batch_size = 1000
batch_size = 2000
batch_size = 3000

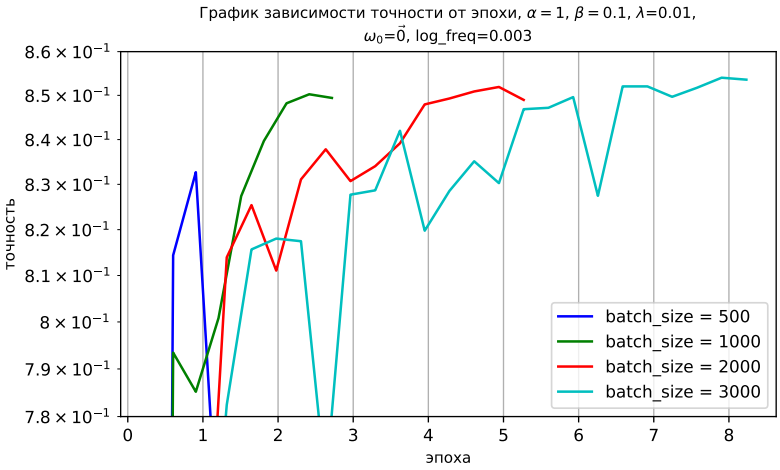


График зависимости точности от эпохи, $\alpha = 1$, $\beta = 0.1$, λ=0.01,
ω_0=$\vec{0}$, log_freq=0.003
точность
эпоха
batch_size = 500
batch_size = 1000
batch_size = 2000
batch_size = 3000