Отчет по второму практическому заданию: градиентные методы обучения линейных моделей. Применение линейных моделей для определения токсичности комментария

Выполнил студент 317 группы, Демьянов Иван

10 ноября 2021 г.

Постановка задания

Данное задание было направлено на ознакомление с линейными моделями и градиентными методами обучения. В нем требовалось реализовать логистическую регрессию при помощи методов градиентного спуска и стохастического градиентного спуска, а также провести ряд экспериментов, направленных на поиск лучших моделей для решения задачи бинарной классификации токсичности комментариев из раздела обсуждений английской Википедии, который был преобразован под следующую задачу: является ли данный комментарий токсичным или нет. Подробнее об исходных данных здесь.

Теоретическая часть

Прежде чем переходить к реализации и экспериментам предлагалось вывести несколько формул, необходимых далее для работы и ввести понятие функции потери логистической регрессии.

Функция потерь бинарной логистической регрессии с L2 регуляризацией имеет следующий вид:

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

Для реализации метода градиентного спуска необходимо вывести $\nabla\mathcal{L}(w)$: сделаем это при помощи матричного дифференцирования, найдя сперва дифференциал функции потерь

$$\mathrm{d}L(w) = \frac{1}{n}\sum_{i=1}^{n}\mathrm{d}(\log(1+\exp^{-y_i\langle x_i,w\rangle})) + \frac{\lambda}{2}\,\mathrm{d}\|w\|^2 = -\frac{1}{n}\sum_{i=1}^{n}\frac{\exp^{-y_i\langle x_i,w\rangle}}{1+\exp^{-y_i\langle x_i,w\rangle}}y_i\langle x_i,\,\mathrm{d}w\rangle + \lambda\,\langle w,\,\mathrm{d}w\rangle$$

Домножив числитель и знаменатель суммы на $\exp^{-y_i\langle x_i,w\rangle}$ и преобразовав дифференциал к каноническому виду получаем следующее:

$$\nabla\mathcal{L}(w) = -\frac{1}{n}\sum_{i=1}^{n}\frac{y_ix_i}{1+\exp^{y_i\langle x_i,w\rangle}} + \lambda w$$

Существует прямое обобщение логистической регрессии на случай многих классов - мультиномиальная регрессия. Пусть построено K линейных моделей $a_1(x),...,a_K(x)$, где $a_j = \mathrm{sign}(\langle w_j, x\rangle)$. Каждая модель дает оценку принадлежности к определенному классу. Эти оценки можно превратить в вероятности с помощью функции $\mathrm{softmax}(z_1,...,z_K)$, которая переводит произвольный вещественный вектор в дискретное вероятностное распределение. Вероятность $k-$го класса можно выразить так:

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

Обучение производится с помощью метода максимального правдоподобия путем минимизации следующей функции потерь:

$$Q(X,w) = -\frac{1}{n}\sum_{i=1}^{n}\log P(y_i|x_i) + \frac{\lambda}{2}\sum_{i=1}^{K}\|w_i\|_2^2$$

Поскольку градиентом в данном случае является матрица, найдем градиент по фиксированному вектору w_m при помощи формул матричного дифференцирования:

$$\frac{\mathrm{d}Q(X,w)}{\mathrm{d}w_m} = \frac{1}{n}\sum_{i=1}^{n}(\frac{\exp^{\langle w_m, x_i\rangle}}{\sum_{j=1}^{K}\exp^{\langle w_j, x_i\rangle}} - [y_i = m])x_i + \lambda w_m$$

В частном случае, при $K = 2$, придем к обычной бинарной логистической регрессии:

$$P(y = 1|x) = \frac{\exp^{\langle w_1, x\rangle}}{\exp^{\langle w_1, x\rangle} + \exp^{\langle w_2, x\rangle}} = \frac{1}{1+\exp^{-\langle w_1 - w_2, x\rangle}}$$

$$P(y = 2|x) = \frac{\exp^{\langle w_2, x\rangle}}{\exp^{\langle w_1, x\rangle} + \exp^{\langle w_2, x\rangle}} = \frac{1}{1+\exp^{\langle w_1 - w_2, x\rangle}} = 1 - P(y = 1|x)$$

Подготовительные эксперименты

Первые два эксперимента можно отнести к подготовительным, так как в них мы анализируем данные и преобразуем их для дальнейшей удобной работы.

В первом эксперименте необходимо было произвести предварительную обработку текста, а именно привести все комментарии к нижнему регистру и заменить в них все символы, не являющиеся буквами и цифрами, на пробелы. При помощи этих изменений в данных остаются важные характеристики, влияющие на токсичность и убираются не особо важные для нашей задачи.

Во втором эксперименте предлагалось преобразовать выборку в разреженную матрицу, где значение x в позиции (i,j) означает, что в документе i слово j встретилось x раз. Это преобразование осуществлялось при помощи класса CountVectorizer из библиотеки sklearn. В таком представлении уже можно работать с данными и оно довольно понятно и удобно. Также в данном эксперименте использовался параметр mindf = 0.001, что позволило существенно уменьшить объем данных, несильно потеряв их свойства, так как были проигнорированы слова, которые очень редко встречаются в документах. Всего различных слов было - 88319, после применения преобразования параметров, указанным выше способом, их стало - 3686, что является заметным уменьшением размерности признакового пространства.

Также на данном этапе было выяснено, что большинство документов в обучающей выборке не я являются токсичными - 35836, количество токсичных комментариев составляет 16225.

Градиентный спуск

В методе градиентного спуска выбирается начальное приближение вектора весов w, затем запускается итерационный процесс, на каждом шаге которого вектор w изменяется в направлении антиградиента функционала $Q(X, w)$:

$$w_{k+1} = w_k - \frac{\alpha}{k^\beta}\nabla Q(w),$$

где α, β - параметры размера шага.

В третьем эксперименте было необходимо исследовать поведение градиентного спуска для задачи логистической регрессии в зависимости от следующих параметров:

- параметр размера шага `step_alpha`
- параметр размера шага `step_beta`
- начального приближения

Данное исследование имеет важную роль, так как оно способствует выявлению лучших гиперпараметров модели. Поскольку в экспериментах меняется только один параметр, а остальные остаются фиксированными, необходимо проанализировать как коррелируют между собой параметры размера шага `step_alpha` и `step_beta`. Ниже приводится зависимость точности предсказаний на валидационной выборке от пар параметров размера шага при 100 итерациях градиентного спуска и коэффициенте регуляризации - 0.1, `step_alpha` рассматривался в диапазоне $[0, 10]$ с шагом 1, `step_beta` - в диапазоне $[0, 2]$ с шагом 0.2. Как можно убедиться из графика, оптимальными являются `step_alpha` из $[7, 10]$, `step_beta` из $[0.8, 1.2]$.

Далее в качестве фиксированных параметров будут использованы $\alpha = 10, \beta = 1$. Начальное приближения будет представлено нулевым вектором признакового пространства, также будут использованы следующие фиксированные параметры: loss_function = 'binary_logistic', tolerance=1e-8, max_iter=100, l2_coef = 0.1.

Рис. 1: График зависимости точности предсказания от гиперпараметров

Исследуем поведение градиентного спуска для задачи логистической регрессии в зависимости от параметра `step_alpha`. Для анализа данной зависимости брались четные `step_alpha` в пределах 10. Графики зависимости функции потерь на обучающей выборке от выбранных `step_alpha` представлены на Рис.2.

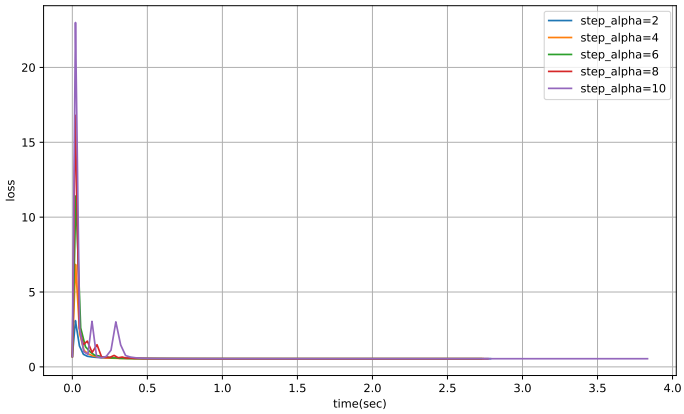

Рис. 2: График зависмости значения функции потерь от реального времени работы метода при 100 итерациях

Как видно по графикам, поначалу функция ошибки на обучающей выборке довольно велика, причем чем больше параметр, тем больше и ошибка. Однако спустя секунду положение улучшается. Несложно видеть, что функция после определенного момента времени не осциллирует, что свидетельствует о хорошем подборе параметров. Можно также отметить вырожденный случай: при `step_alpha`=0 градиент зануляется, веса остаются такими же, следовательно функция потерь не изменяется, поэтому алгоритм прекращает свою работу, значит такое значение является худшим. Если значение `step_alpha`близко к нулю, то может произойти аналогичное и роль градиента в формировании нового вектора весов нивелируется.

Также в ходе данного эксперимента исследовалась зависимость и при большем количестве итераций: графики

были схожи на текущие графики по виду, поэтому было принято решение оставить самый удачный из них по виду.

Аналогичными предыдущим получились графики зависимости функции потерь от итерации метода. Дабы не загромождать место такими же по виду графиками, они размещены в приложениях к отчету. Аналогичность обусловлена почти одинаковыми интервалами времени от итерации.

Ниже представлены графики зависимости точности (accuracy) на валидационной выборке от реального времени работы метода. Валидационная выборка составляет 30% от изначальной.

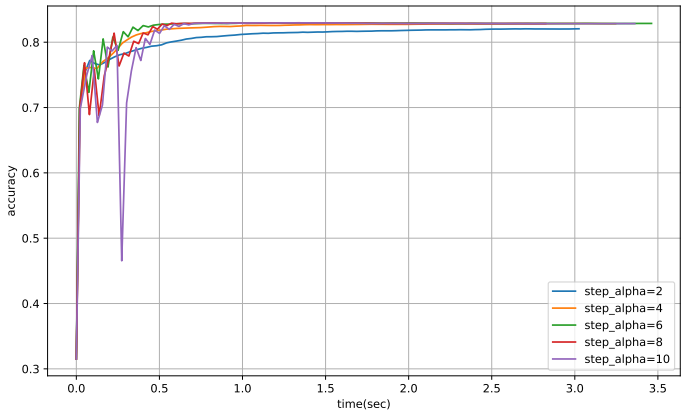

Рис. 3: График зависимости точности (accuracy) на валидационной выборке от реального времени работы метода при 100 итерациях

Отметим, что если "перевернуть"данные графики, то получатся очень похожие графики на Рис.2. Другими словами, функция потерь и функция точности предсказания являются обратно пропорциональными, что вполне логично. График зависимости точности (accuracy) итерации метода является аналогичным, поэтому из тех же соображений экономии места, приведен в приложении к отчету.

Теперь перейдем к исследованию поведения градиентного спуска для задачи логистической регрессии в зависимости от параметра `step_beta`. Графики зависимости функции потерь на обучающей выборке от выбранных `step_beta` приведены ниже.

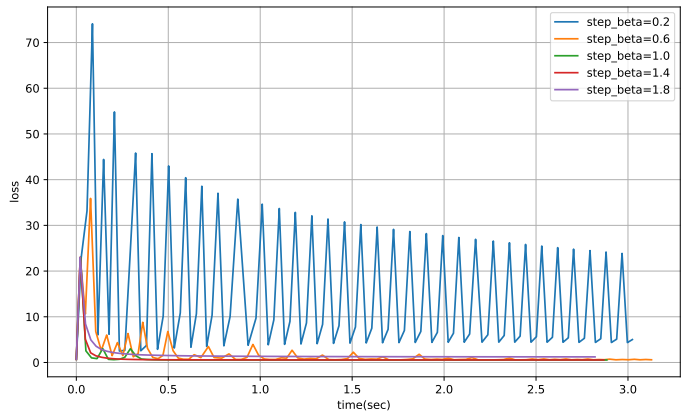

Рис. 4: График зависимости значения функции потерь от реального времени работы метода при 100 итерациях

Благодаря этому графику можно понять, что модели, в которых шаг градиентного спуска достаточно мал (`step_beta` < 0.4) дольше сходятся, а модели, в которых шаг не зависит от номера итерации (`step_beta` $= 0$) неустойчивы (график функции потерь от нулевого `step_beta` приведен в приложении, так как накладывался

на другие график за счет осцилляций). Также можно отметить, что значения из предварительно отобранного диапазона для данного гиперпараметра очень хорошо себя показали. Модели, в которых параметр больше указанного выше оптимального диапазона устойчивы, но плохи тем, что коэффициент перед градиентом начинает слишком быстро стремиться к нулю из-за чего градиентный спуск не успевает сойтись к оптимуму и предсказания получаются неточными.

Аналогичный вид имеют и графики зависимости функции потерь от итерации метода, которые приведены в приложениях к отчету. Рассуждения про оставшиеся графики, описывающие поведение градиентного спуска для задачи логистической регрессии в зависимости от параметра `step_beta`, такие же как и у `step_alpha`, поэтому тут будет приведен график зависимости точности (accuracy) на валидационной выборке от реального времени работы метода при 100 итерациях, а аналогичный график зависимости от итерации - в приложении к отчету.

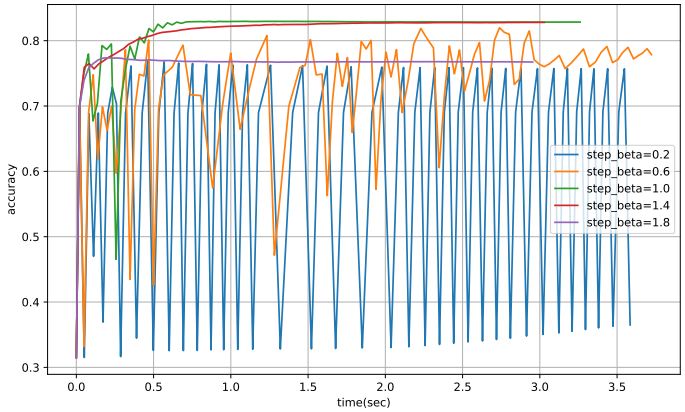

Рис. 5: График зависимости точности (accuracy) на валидационной выборке от реального времени работы метода при 100 итерациях

До этого момента вектор w_0 инициализировался нулями. Рассмотрим его инициализацию при помощи вероятностных распределений и сравним с текущей. Ниже приведены зависимости функции потерь от стратегий заданий начального приближения. Здесь параметр `step_alpha` был заменен на 2, вместо 10, так как при больших значениях наблюдаются большие скачки на первых итерациях.

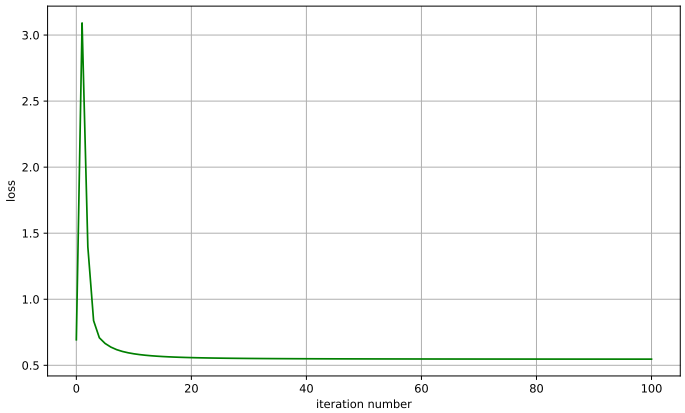

Рис. 6: График зависимости функции потерь от реального времени работы метода при 100 итерациях при инициализации w_0 нулями

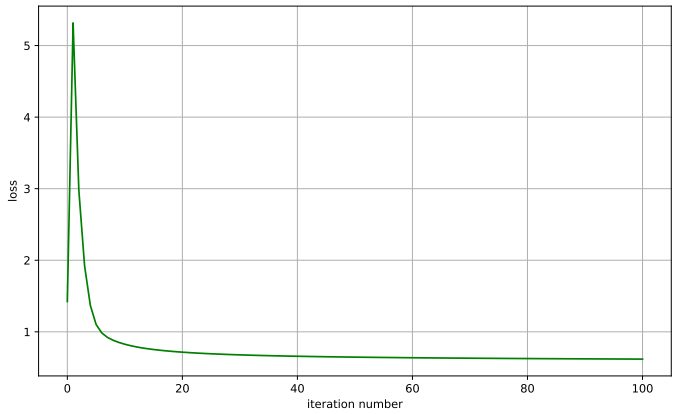

Рис. 7: График зависимости функции потерь от реального времени работы метода при 100 итерациях при инициализации w_0 при помощи равномерного распределения с параметрами [-1, 1]

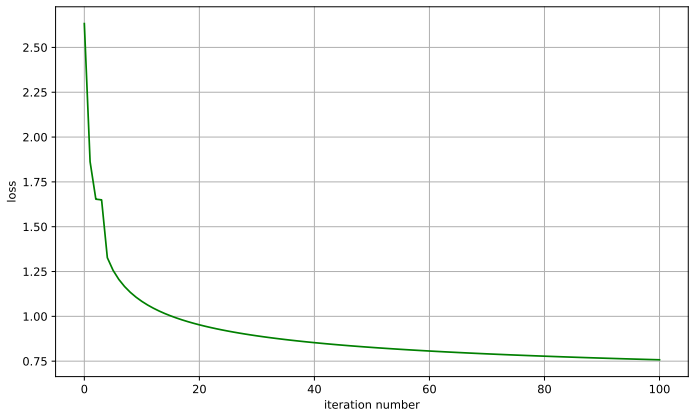

Рис. 8: График зависимости функции потерь от реального времени работы метода при 100 итерациях при инициализации w_0 при помощи нормального распределения с параметрами [0, 0.1]

Судя по графикам, стратегия инициализации при помощи равномерного распределения является неудачной. Причиной этому может являться то, что большинство слов в документах несут в себе нейтральный характер, следовательно их веса должны иметь близкое к нулю значение. Учитывая вышесказанное, приходим к тому, что лучшей стратегией оказывается инициализация при помощи нормального распределения с параметрами [0, 0.1]. Эта стратегия является лучшей, потому что на графике отсутствуют скачки. Скачки могут свидетельствовать о том, что начальное приближение далеко от оптимальных весов, следовательно алгоритму пришлось идти к ним через точки, на которых велико значение ошибки.

По другим зависимостям можно сделать аналогичные выводы, они приведены в приложении.

Стохастический градиентный спуск

Данный метод отличается от предыдущего тем, что градиент на каждой итерации вычисляется не по всей выборке, а по ее батчам, которые генерируются в течении эпохи. Такой подход позволяет сэкономить в вычислениях и при правильно подобранных параметрах является очень выгодным. Проведем эксперименты по аналогии с градиентным спуском.

Исследуем как размер батча влияет на функцию потерь и точность при раннее подобранных параметрах. Прежде всего, отметим, что при маленьких в соотношении с размером выборки размерах батча не будет об-

новлений, так как приближенный номер эпохи будет меньше, чем параметр частоты обновления, который по умолчанию равен 1.

Исходя из этой причины в экспериментах будут браться следующие размеры батча из диапазона $[1000, 10000]$ с размером шага - 2000.

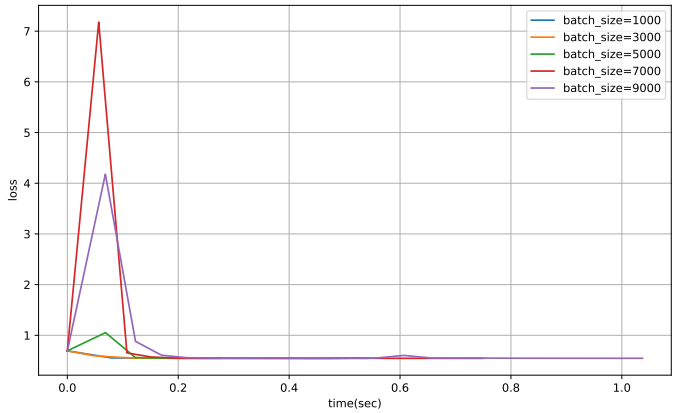

Рис. 9: График зависимости значения функции потерь от реального времени работы метода при 100 итерациях

Избежать проблему выше можно и за счет увеличения параметра максимального числа итераций до 1000. Ранее он был 100, так как вид графиков был приемлемым и не менялся в течении последующих итераций. В этом эксперименте будем использовать размеры батча от 200 до 1000 с шагом 200.

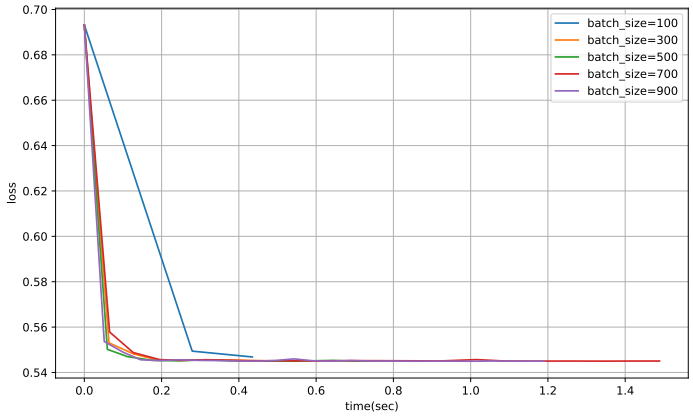

Рис. 10: График зависимости значения функции потерь от реального времени работы метода при 1000 итерациях

Демонстрация некоторых графиков менее информативна, так как зависит от батча и вполне логично: чем меньше батч, тем меньше эпох, на которых мы производим замер характеристик модели. Более удачно себя зарекомендовал размер батча 5000 при 100 итерациях, поэтому далее будем рассматривать его в качестве фиксированного параметра. Также следует добавить, что графики зависимости функции потерь по итерации выглядят аналогично, найти их можно в приложении к отчету.

Далее рассмотрим зависимость точности на валидационной выборке от размера батча. Как видно по графи-

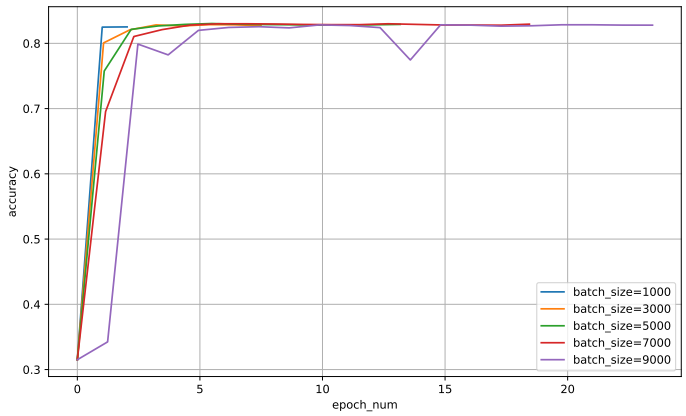

Рис. 11: Зависимость точности от эпохи

кам, отсутствуют большие скачки в отличии от градиентного спуска. Графики стали более плавными. Можно предположить, что это достигается за счет того, что если мы на какой-то итерации и далеко от оптимума, то приближения проводится не на всей выборке, сопроводжая при этом большую ошибку у функции потерь, а постепенно на какой-то части. По аналогии с предыдущими пунктами аналогичный график от времени находится в приложении.

Аналогично экспериментам прошлого пункта, посмотрим как от `step_alpha` зависит поведение стохастического градиентного спуска. Ниже приведены 2 графика, аналогичные им находятся в приложении.

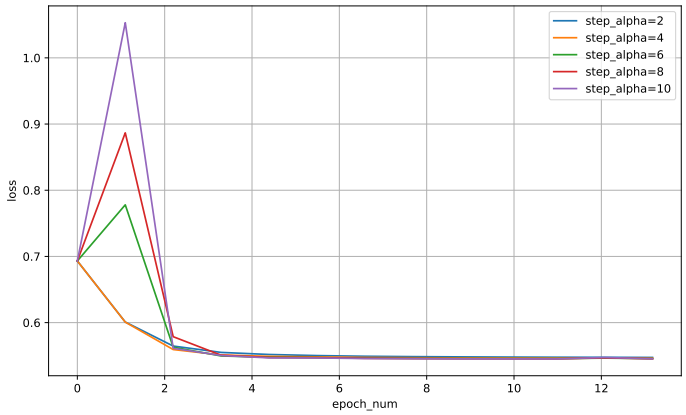

Рис. 12: Зависимость функции потерь от эпохи

Выводы из графиков такие же как и в случае аналогичного эксперимента с градиентным спуском: при больших значениях параметра замечается более сильный скачок графика в начале, однако теперь этот скачок стал менее сильным по сравнению с обычным градиентным спуском.

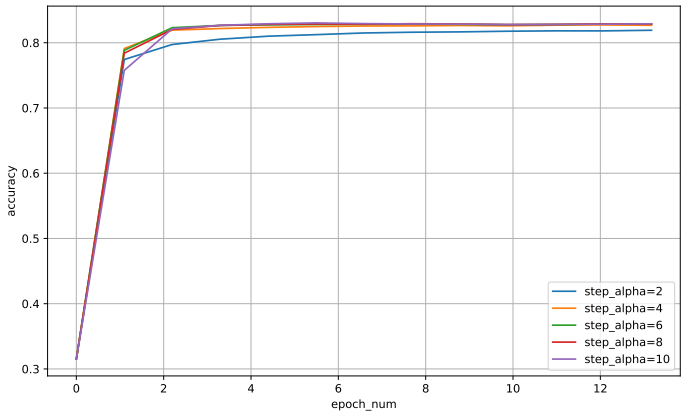

Рис. 13: Зависимость точности от эпохи

Перейдем теперь к экспериментам с параметром `step_beta`. Ниже представлены графики зависимостей, характеризующих влияние параметра `step_beta` на поведение стохастического градиентного спуска.

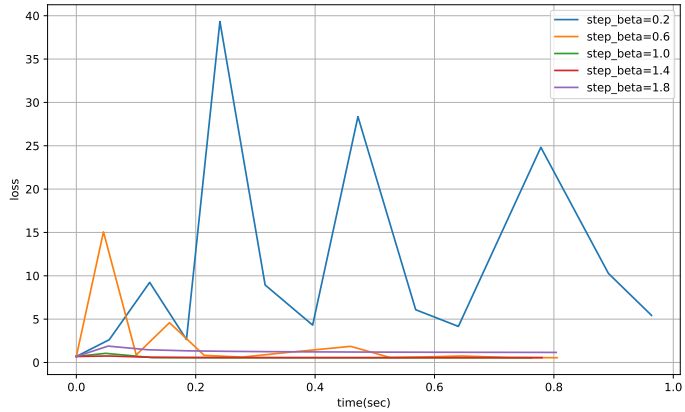

Рис. 14: Зависимость функции потерь от времени

Выводы из графиков такие же как и в случае аналогичного эксперимента с градиентным спуском: при слишком малом значении параметра алгоритм является неустойчивым, у моделей с большим значением параметра тоже имеется недостаток: они устремляют шаг в направлении антиградиента к нулю, что может привести к тому, что алгоритм будет не успевать сходиться к оптимуму и, как следствие, давать неточные предсказания. Также стоить отметить, что графики стали более плавными.

Общий касательно параметров `step_alpha` и `step_beta` таков, их необходимо подбирать так, чтобы коэффициент при антиградиенте был не слишком маленьким, иначе градиентный спуск/стохастический градиентный спуск теряет смысл.

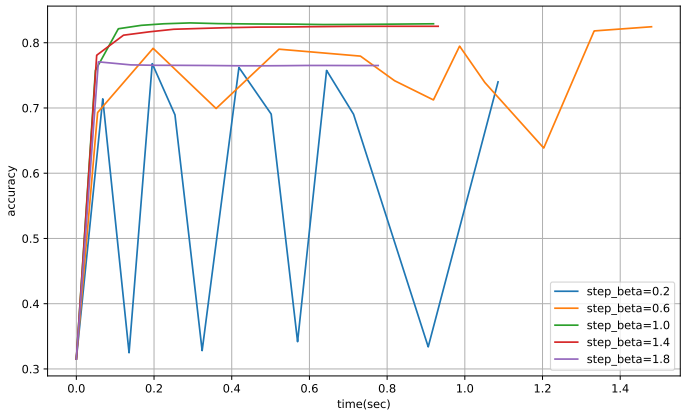

Рис. 15: Зависимость точности от времени

Ну и наконец, исследуем поведение модели в зависимости от стратегий заданий начального приближения. Стратегии задания начального приближения такие же как и в аналогичном эксперименте прошлого пункта. Ниже приведены 2 графика, описывающие результаты данного эксперимента, 2 аналогичных графика находятся в приложении к отчету.

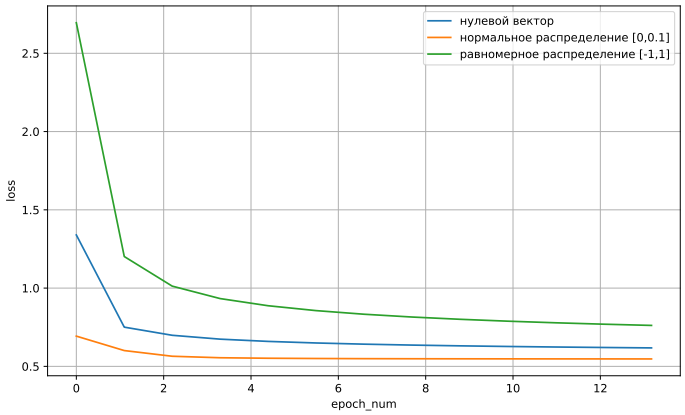

Рис. 16: Зависимость функции потерь от эпохи

Подводя вывод к данному эксперименту, можно отметить, что результаты в плане качества стратегий совпали с аналогичным экспериментом прошлого пункта, однако за счет того, что мы работаем на первых итерациях не со всей выборкой, а с ее батчем, даже начальное приближение, которое далеко от точки оптимума, не будет иметь сильных скачков, так как приближение к оптимуму будет производится плавно и функции не придется сразу переходить к другой точке ценой большой функции ошибки.

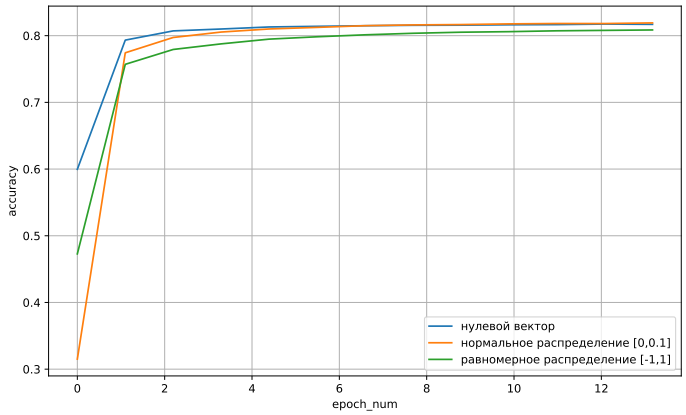

Рис. 17: Зависимость точности от эпохи

Сравнение градиентного спуска и стохастического градиентного спуска

В предыдущих экспериментах мы рассматривали различные зависимости, связанные с этими моделями, попутно сравнивая их. Сравним работу ГС и СГС и подведем окончательный выводы касательно этих моделей.

Здесь стоит упомянуть, что количество объектов, обработанных за эпоху при помощи СГС совпадает с количеством объектов, обработанных за одну итерацию при помощи ГС. Из этого следует, что сравнение методов по итерациям некорректно! Учтя вышесказанное, будем сравнивать работу методом по времени и по эпохам (эпоха в ГС совпадает с итерацией). Построим графики, описывающие эти сравнения.

Ниже будут приведены зависимости от эпох. Зависимости от времени получились аналогичными, найти их можно в приложении.

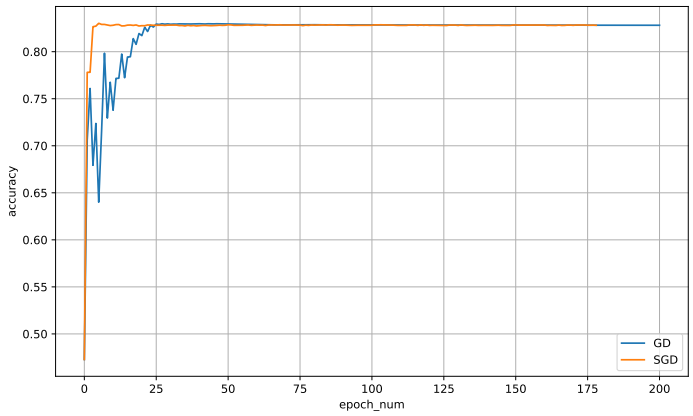

Рис. 18: Зависимость функции потерь от эпохи

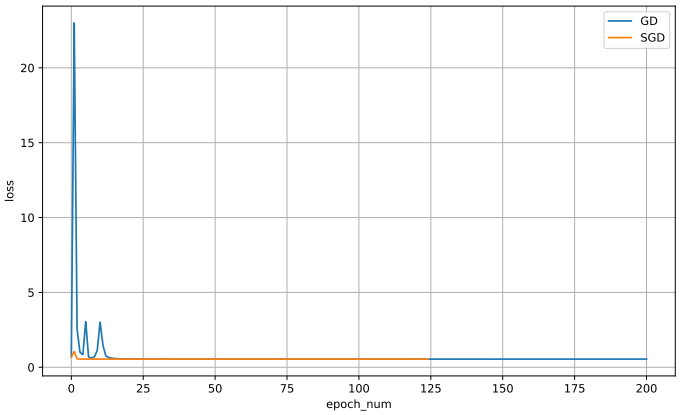

Рис. 19: Зависимость точности от эпохи

Итог сравнения GD и SGD: стохастический градиентный спуск является общим случаем градиентного спуска. Градиентный спуск может быть получен из. стохастического путем задания размера батча, равному размеру выборки. Таким образом, SGD является более гибким методом за счет дополнительного параметра в виде размера батча. В случае удачного подбора размера батча получаются довольно плавные, как на графиках выше, зависимости функции потерь и точности. К преимуществам SGD также относится отсутствие необходимости хранения в оперативной памяти всей выборки.

Лемматизация

Лемматизация — процесс приведения слова к лемме — его нормальной (словарной) форме. Для лематизации наших данных применим алгоритм лемматизации (WordNetLemmatizer из библотеки nltk) к коллекции документов. Также удалим из текста стоп-слова (используя список стоп-слов из nltk).

Очевидно, что после предобработки корпуса таким образом количество слов в тексте будет уменьшено, следовательно размерность признакового пространства будет сокращена. С учетом этого сократится и время обучения модели. При параметре $min\_df = 0.001$ количество признаков сократилось с 3686 по 3350.

Далее были рассмотрены следующие модели и точности на них:

- модель - SGDClassifier(loss_function = 'binary_logistic', batch_size=5000, step_alpha=10, step_beta=1, tolerance=1e-8, max_iter=1000, l2_coef=0.1)
  - точность с лемматизацией: 0.832
  - точность без лемматизации: 0.827
- модель - SGDClassifier(loss_function = 'binary_logistic', batch_size=5000, step_alpha=2, step_beta=0, tolerance=1e-8, max_iter=1000, l2_coef=0)
  - точность с лемматизацией: 0.897
  - точность без лемматизации: 0.846
- модель - GDClassifier(loss_function = 'binary_logistic', step_alpha=10, step_beta=1, tolerance=1e-8, max_iter=100, l2_coef = 0.001)
  - точность с лемматизацией: 0.858
  - точность без лемматизации: 0.833

Таким образом, видим, что точность моделей при лемматизации увеличилась.

Способы векторного представления текста

До сих пор, чтобы преобразовать текст в данные, на которых можно обучаться, мы использовали представление BagOfWords. Оно возвращало разреженную матрицу, в которой значение x в позиции (i,j) означает, что в документе i слово j встретилось x раз.

В противоположность этому подходу можно использовать представление Tfidf. Tfidf — статистическая мера, используемая для оценки важности слова в контексте документа, являющегося частью коллекции документов. Вес некоторого слова пропорционален частоте употребления этого слова в документе и обратно пропорционален частоте употребления слова во всех документах коллекции.

Отметим, что признаковое пространство у вышеперечисленных моделей одинаковое, значит время тоже приблизительно одинаковое. Разнится лишь информация, которая хранится в матрицу признаков, следовательно существенную роль сравнения играет точность.

Сравним точности предсказаний этих методов на валидационной выборке, используя модель с параметрами SGDClassifier(loss_function = 'binary_logistic', batch_size=5000, step_alpha=10, step_beta=1, tolerance=1e-8, max_iter=1000, l2_coef=0.001):

- точность с BagOfWords: 0.848
- точность с Tfidf: 0.741

Как видим, модель представления BagOfWords показала лучшие результаты, но это сравнение не совсем честно: в ходе с параметрами GD и SGD были подобраны параметры с учетом представления BagOfWords.

Так, например, модель SGDClassifier(loss_function = 'binary_logistic', batch_size=5000, step_alpha=10, step_beta=0, tolerance=1e-8, max_iter=1000, l2_coef=0.0001) показала следующие результаты:

- точность с BagOfWords: 0.813
- точность с Tfidf: 0.885

Исходя из этого можно предположить, что модели с представлением Tfidf работают лучше, когда нивелируется порядок итерации. Также в выводе этого эксперимента стоит отметить, что у двух представлений есть свои преимущества, для раскрытия которых стоит, вообще говоря, подбирать разные параметры.

Также следует отметить, что у конструкторов, с помощью которых реализованы представления, есть параметры min_df и max_df. Первый из них служит для отбрасывания редко встречающихся слов в словаре, второй - часто встречающихся. До этого момента min_df был равен 0.001. Проведем эксперименты со следующей моделью: SGDClassifier(loss_function = 'binary_logistic', batch_size=5000, step_alpha=2, step_beta=0, tolerance=1e-8, max_iter=1000, l2_coef=0.0001). Результаты экспериментов приведены ниже в таблице 1.

min_df	max_df	dim	Accuracy
_	_	88319	0.887
0.001	_	3686	0.879
_	0.001	84633	0.714
_	0.5	88316	0.9
0.001	0.5	3683	0.895
0.1	_	55	0.498

Таблица 1: Результаты экспериментов на валидационной выборке

По результатам экспериментов можно сделать следующие выводы:

- необязательно брать в признаковое пространство все слова, так как модель может переобучаться на шумы
- не стоит обучаться только на редких признаках, это ухудшает модель из-за недостаточного количества ответов на объектах, в которых они присутствуют(эксперимент 3)
- можно довольно сильно уменьшить количество признаков, не особо потеряв качество, но значительно ускорив алгоритм
- сокращение признакового пространства способно увеличить точность(эксперимент 4)
- слишком малая размерность чревата большим количеством ошибок(последний эксперимент), поэтому следует искать 'золотую середину' между уменьшением размерности признакового пространства и точностью (эксперимент 2).

Применение алгоритма к тестовой выборке и анализ ошибок

К тестовой выборке была применена следующая модель: SGDClassifier(loss_function = 'binary_logistic', batch_size=5000, step_alpha=2, step_beta=0, tolerance=1e-8, max_iter=1000, l2_coef=0.0001)

Точность: 0.852.

Ниже приведена матрица ошибок. Как видим, большинство ошибок достигнуто за счет того, что модель ошибочно считала комментарий токсичным.

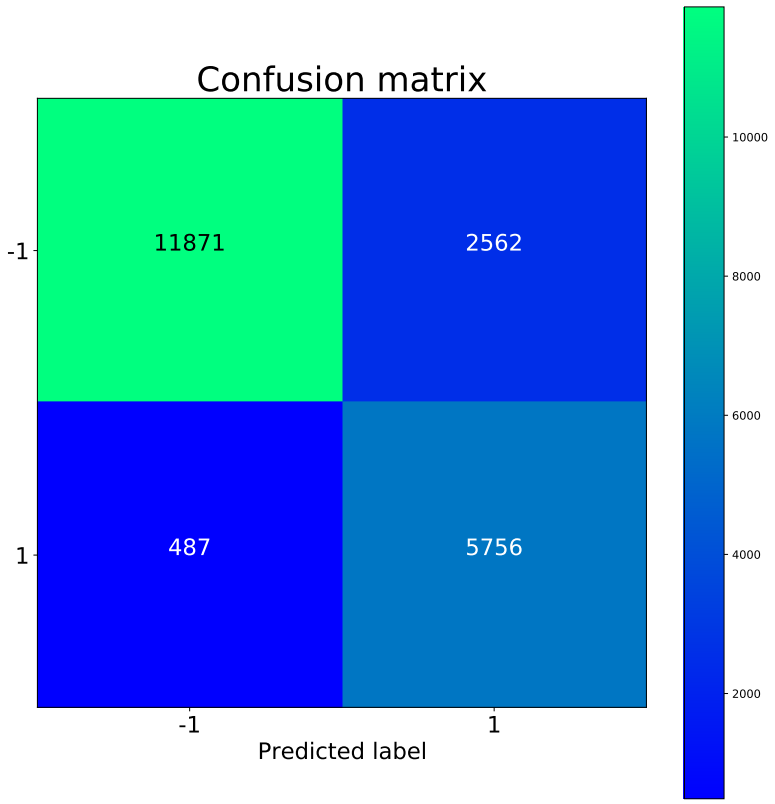

Рис. 20: Матрица ошибок

Модели дают неправильные ответы в основном в случае употребление жаргона и разговорной речи. Например, в данном комментарии

'"::seems to be more than the usual amount of idiocy around - LOL! ::a) At various times, Iv́e made similar observations. i.e. You are by NO means an orphan in this respect. ::b) Actually, I suspect the usual amount of idiocy is fairly constant. I suspect the variable is your and my perception and/or level-of-tolerance of it ... ::Andreas is a difficult person to agree with - Andreas is a difficult person. ::Iv́e been a little worried about my comments in various places being taken offence to. - In my biased opinion, I see that as a good thing; Id́ be more worried if (like some) you always thought you were right. ::Cheers, "
слово 'idiocy' было употреблено, чтобы описать нелепость ситуации, а не в целях оскорбления.

Общие выводы

В данном отчете были представлены результаты экспериментов с логистической регрессией. Начало отчета занимала постановка задачи и теоретическая часть. В ходе экспериментов было произведено сравнение градиентного спуска и стохастического. градиентного спуска, подобраны для них параметры и изучены зависимости метода от каждого из параметров. На примере задачи бинарной классификации были рассмотрены методы работы с текстовыми данными, такие как BagOfWords и Tfidf. Также в ходе работы было проведено исследование методов сокращения признакового пространства за счет лемматизации, стоп-слов и других параметров. Ну и наконец, модель была применена к тестовой выборке и был проведен анализ самых частых ее ошибок.

Литература

Семинарский конспект по теме "Линейные модели для классификации".

Приложение

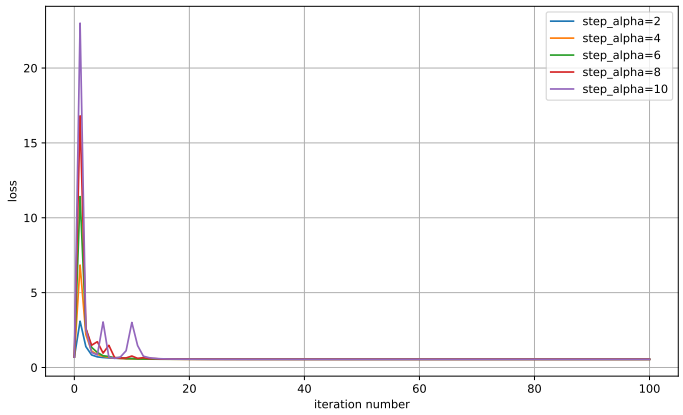

Рис. 21: График зависимости значения функции потерь от итерации метода при изменении alpha

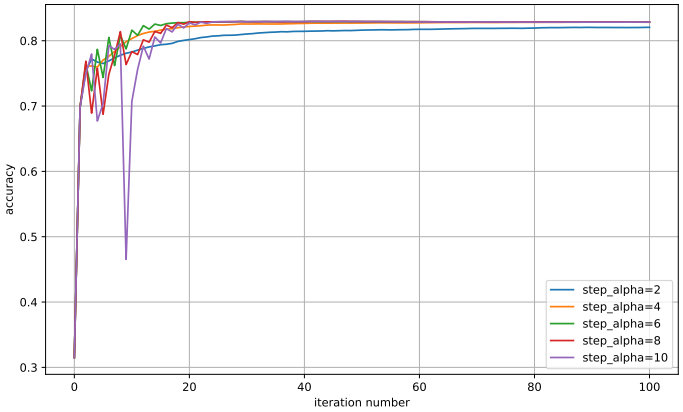

Рис. 22: График зависимости точности (accuracy) итерации метода при изменении alpha

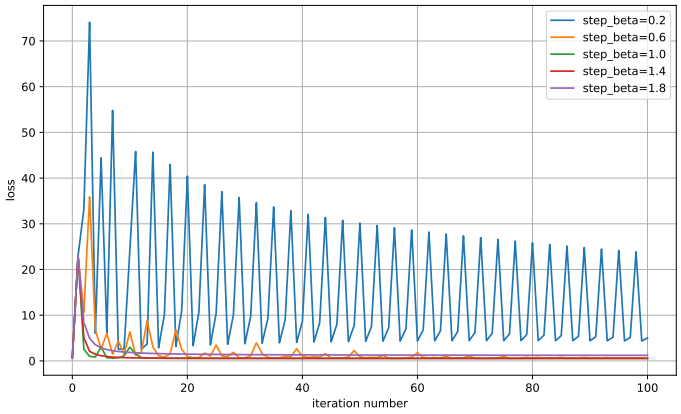

Рис. 23: График зависимости значения функции потерь от итерации метода при изменении beta

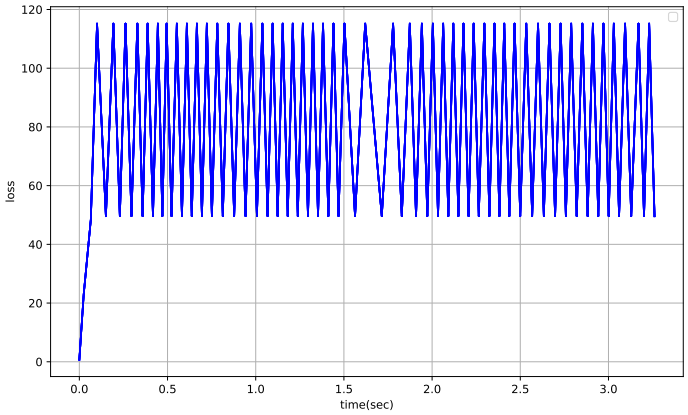

Рис. 24: График зависимости значения функции потерь от итерации метода при beta=0

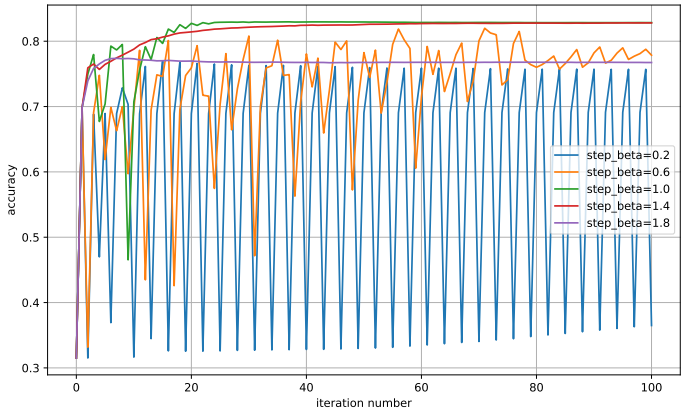

Рис. 25: График зависимости точности (accuracy) итерации метода при изменении beta

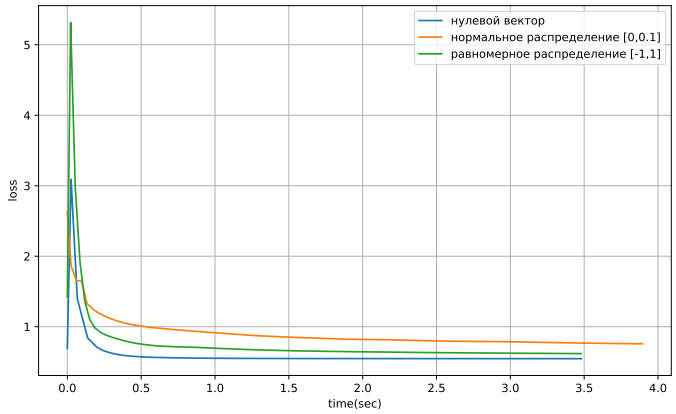

Рис. 26: Зависимость функции потерь при различных инициализациях w_0

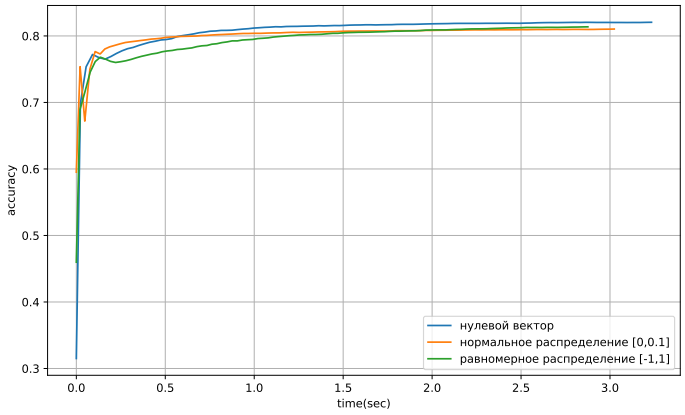

Рис. 27: Зависимость точности от времени при различных инициализациях w_0

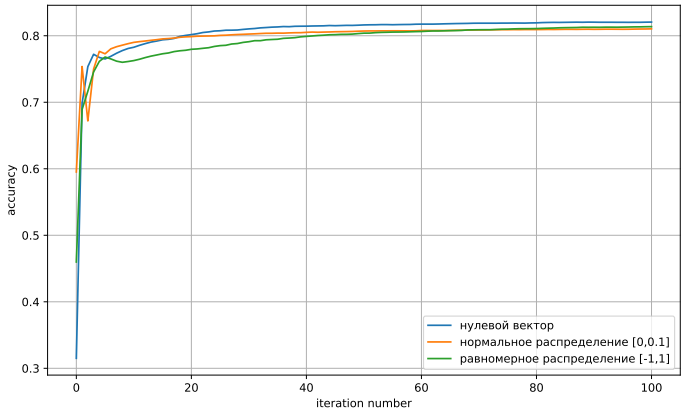

Рис. 28: Зависимость точности от итерации при различных инициализациях w_0

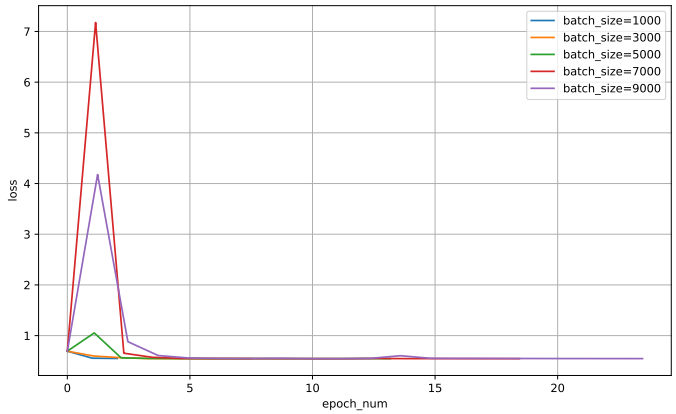

Рис. 29: График зависимости функции потерь от эпохи, на которой делается замер, при 100 итерациях

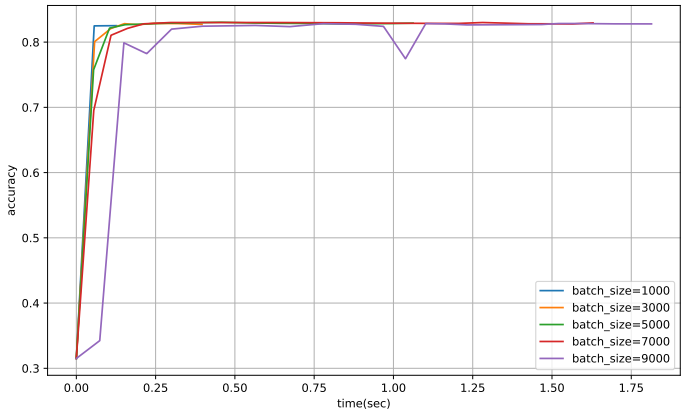

Рис. 30: Зависимость точности от времени

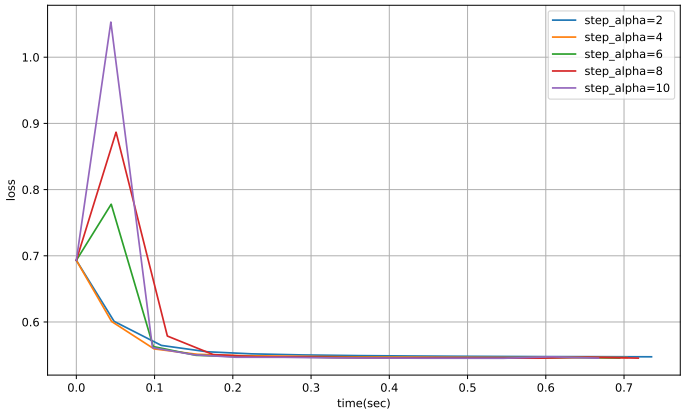

Рис. 31: Зависимость функции потерь от времени

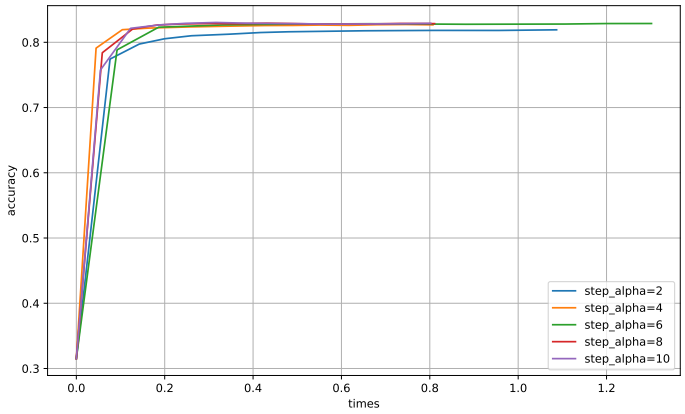

Рис. 32: Зависимость точности от времени

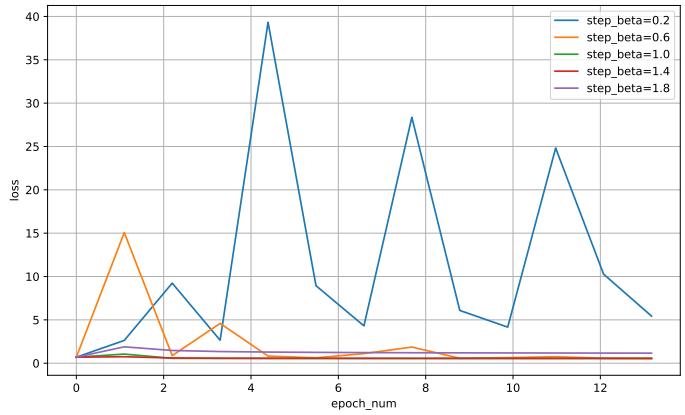

Рис. 33: Зависимость функции потерь от итерации

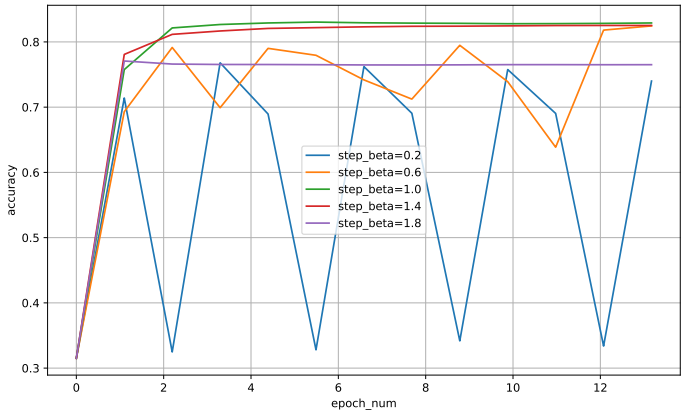

Рис. 34: Зависимость точности от итерации

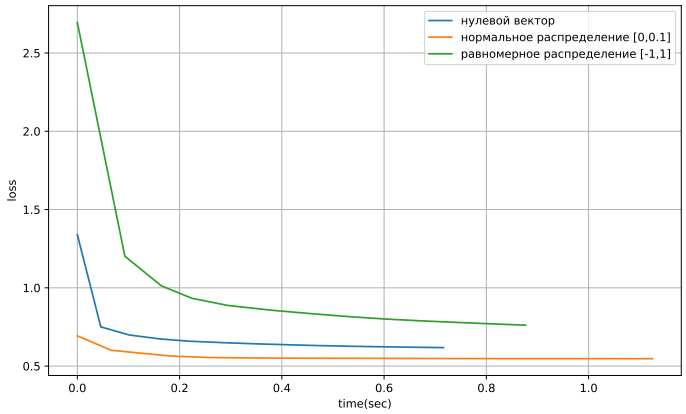

Рис. 35: Зависимость функции потерь от времени

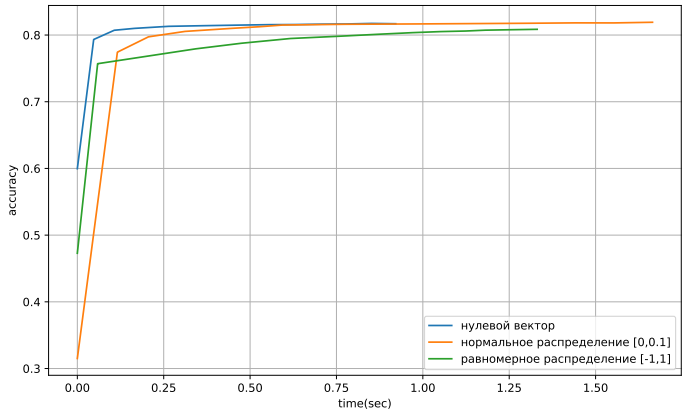

Рис. 36: Зависимость точности от времени

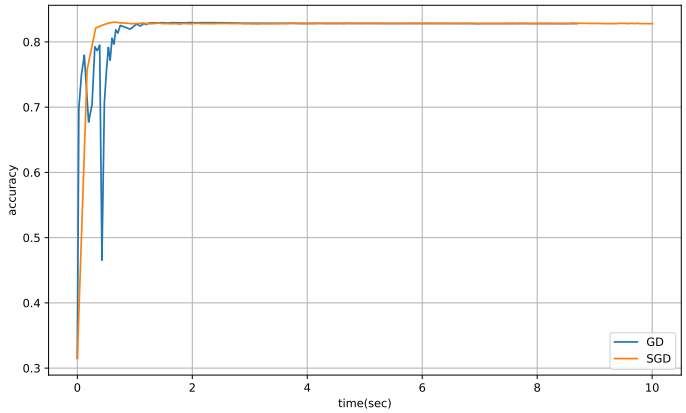

Рис. 37: Зависимость функции потерь от времени

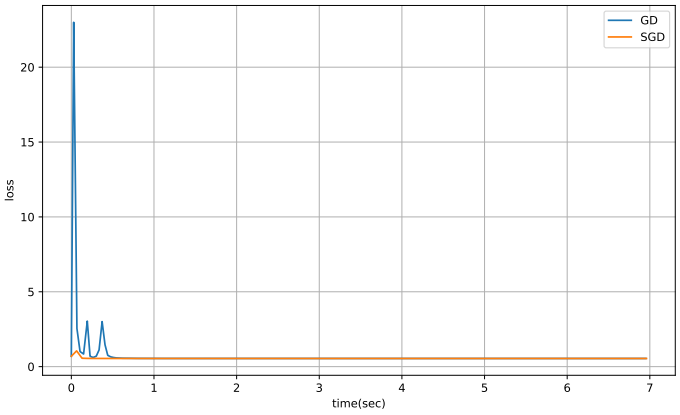

Рис. 38: Зависимость точности от времени