Московский Государственный Университет им. М. В. Ломоносова
Факультет вычислительной математики и кибернетики

Отчёт по заданию «Градиентные методы обучения линейных моделей»

М. М. Шамшиев
317 группа
27 ноября 2017

Аннотация

В данном отчете отражены основные результаты, полученные в ходе выполнения практического задания «Градиентные методы обучения линейных моделей». Целью задания было написание собственной реализации линейного классификатора с логистической функцией потерь, а также многоклассовых классификаторов one-vs-all и all-vs-all. Также было необходимо провести ряд экспериментов на наборах данных real-sim [1] и 20newsgroups [2] и сделать выводы, основываясь на полученных результатах.

Содержание

1 Теоретическая часть

Пусть $X \in \mathbb{R}^{l \times d}$ — матрица объекты-признаки, $y \in \mathbb{R}^l$ — вектор ответов, $\omega \in \mathbb{R}^d$ — вектор весов, $\lambda \in \mathbb{R}$ — коэффициент регуляризации. Обозначим через $e = (1, \ldots, 1)^T \in \mathbb{R}^l$ вектор-столбец, состоящий из единиц, а через $\sigma(M) = 1 \Big/ (1 + e^{-M})$ — сигмоидную функцию. Также под применением скалярной функции к вектору будем понимать применение данной функции к каждому элементу вектора. Тогда формулы для задачи логистической регрессии и ее градиента представимы в виде:

$$Q(X, \omega) = \frac{1}{l} e^T \log\left(e + \exp(-y \odot X\omega)\right) + \frac{\lambda}{2}\|\omega\|_2^2$$

$$\nabla_\omega Q(X, \omega) = -\frac{1}{l} X^T (y \odot \sigma(-y \odot X\omega)) + \lambda\,\omega$$

Пусть теперь $\omega \in \mathbb{R}^{K \times d}$ — набор весов, где K - количество классов, а через $[y == j]$ обозначим индикатор того, что число y равно j. В таких обозначениях формулы для задачи мультиномиальной регрессии и ее градиента записываются следующим образом:

$$Q(X, \omega) = -\frac{1}{l} \sum_{i=1}^{l} \langle \omega_{y_i},\, x_i \rangle + \frac{1}{l} \sum_{i=1}^{l} \log \left(\sum_{k=1}^{K} \exp(\langle \omega_k,\, x_i \rangle) \right) + \frac{\lambda}{2} \sum_{k=1}^{K} \|\omega_k\|_2^2$$

$$\nabla_\omega Q(X, \omega) = \left(\frac{\partial Q}{\partial \omega_1}, \ldots, \frac{\partial Q}{\partial \omega_K} \right),$$

$$\frac{\partial Q}{\partial \omega_j} = -\frac{1}{l} \sum_{i=1}^{l} x_i \,[y_i == j] + \frac{1}{l} \sum_{i=1}^{l} \frac{x_i \exp(\langle w_j,\, x_i \rangle)}{\sum_{k=1}^{K} \exp(\langle \omega_k,\, x_i \rangle)} + \lambda\,\omega_j$$

Покажем, что при $k = 2$ задача многоклассовой логистической регрессии сводится к бинарной логистической регрессии. Пусть метки классов принадлежат множеству $\{+1, -1\}$. Запишем функционал, соответствующий задачи мультиномиальной регрессии и преобразуем его:

$$Q(X, \omega) = -\frac{1}{l} \sum_{i=1}^{l} \log P(y_i | x_i) = -\frac{1}{l} \sum_{i=1}^{l} \log \left(\frac{\exp(\langle \omega_{y_i},\, x_i \rangle)}{\exp(\langle \omega_{+1}, x_i \rangle) + \exp(\langle \omega_{-1},\, x_i \rangle)} \right) =$$

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

где $\omega = \omega_{+1} - \omega_{-1}$.

Полученный функционал совпадает с минимизируемым функионалом задачи бинарной логистической регрессии.

2 Сравнение алгоритмов полного градиентного спуска и стохастического градиентного спуска для задачи логистической регрессии

Сравнение работы алгоритмов проводилось на наборе данных real-sim. В соответствии с заданием, были использованы следующие параметры моделей:

$l2\_coef = 10^{-5}$, $tolerance = 10^{-5}$, $step\_alpha = 1$, $step\_beta = 0$, $batch\_size = 1$

Также для полного градиентного спуска был выбран параметр $max\_iter = 10000$, а для стохастического — $max\_iter = 100000$.

Зависимости значений функций потерь от времени работы методов отображены на Рис. 1 и 2, а от номера итерации и эпохи — на Рис. 3 и 4.

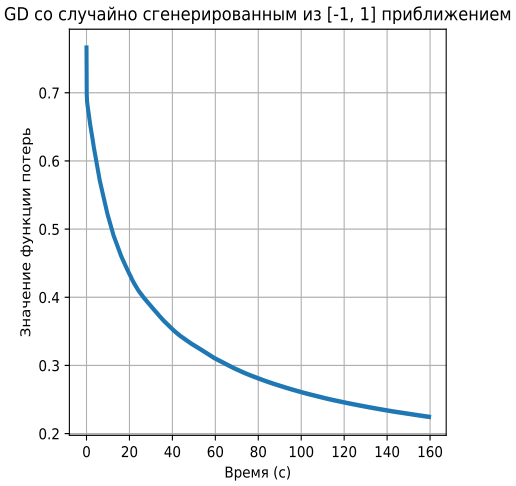

Рис. 1: Зависимость значения функции потерь от времени работы полного градиентного спуска

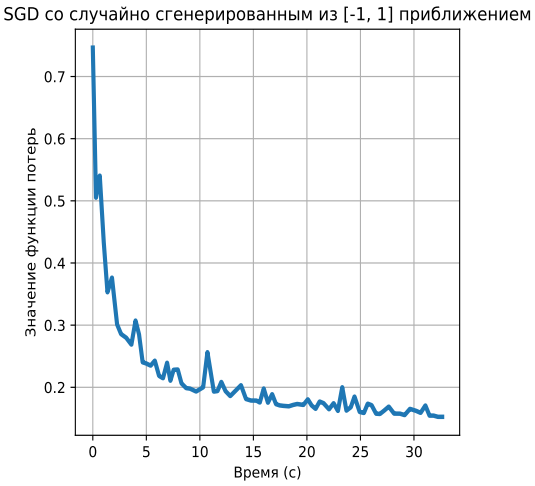

Рис. 2: Зависимость значения функции потерь от времени работы стохастического градиентного спуска

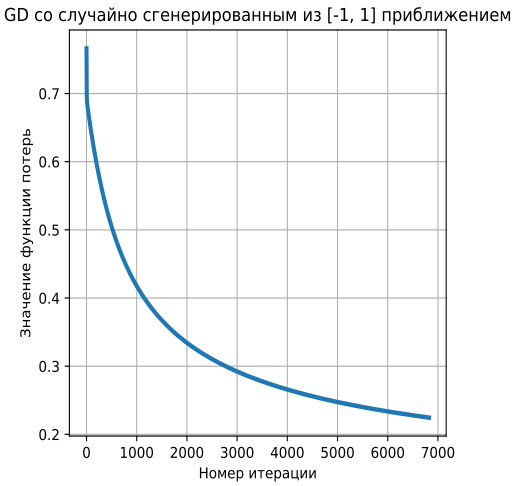

Рис. 3: Зависимость значения функции потерь от номера итерации полного градиентного спуска

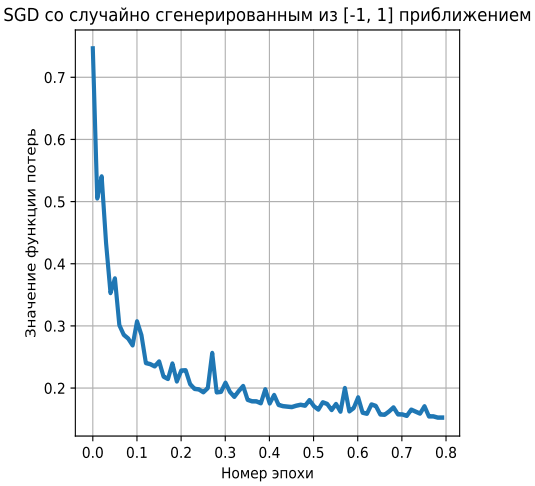

Рис. 4: Зависимость значения функции потерь от номера эпохи стохастического градиентного спуска

Из приведенных графиков видно, что при выбранных параметрах оба метода достаточно хорошо минимизируют функцию потерь. Из Рис. 3 можно заметить, что

количество выполненных итераций полного градиентного спуска меньше установленного параметра max_iter, а значит метод вышел по критерию остановки, задаваемого параметром $tolerance$. В то же время из Рис. 4 можно сделать вывод, что метод стохастического градиентного спуска также закончил работу досрочно, поскольку приближенный номер эпохи не соответствует выбранному значению параметра max_iter. В конечном счете время обучения модели с помощью полного градиентного спуска превысило соответствующее время для стохастического примерно в 5 раз. Более того, стохастический градиентный спуск достиг меньшего значения функционала, чем полный.

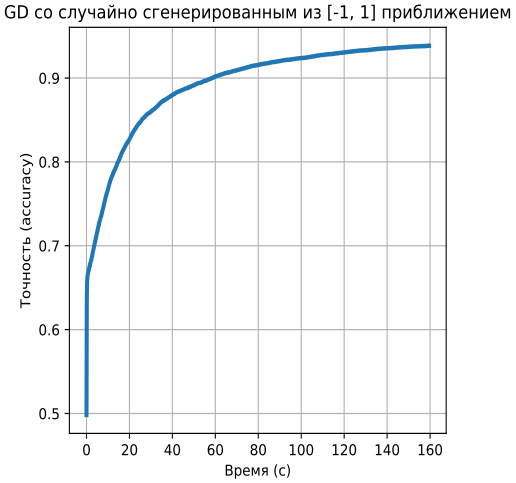

Рис. 5: Зависимость точности от времени работы полного градиентного спуска

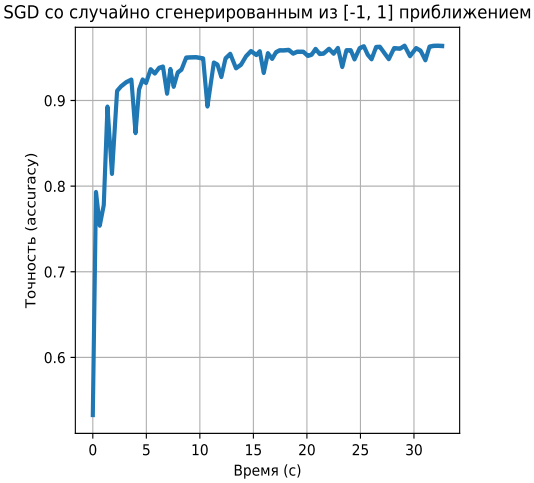

Рис. 6: Зависимость точности от времени работы стохастического градиентного спуска

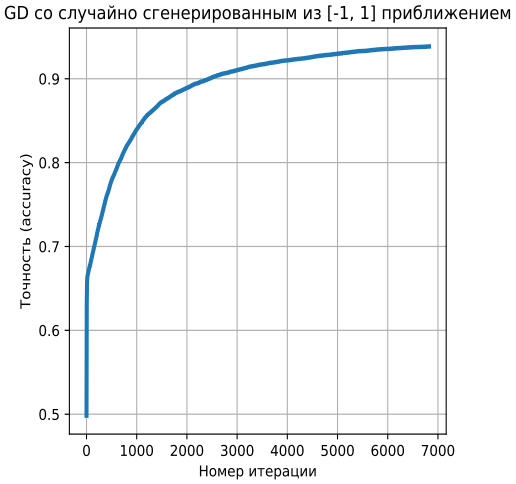

Рис. 7: Зависимость точности от номера итерации полного градиентного спуска

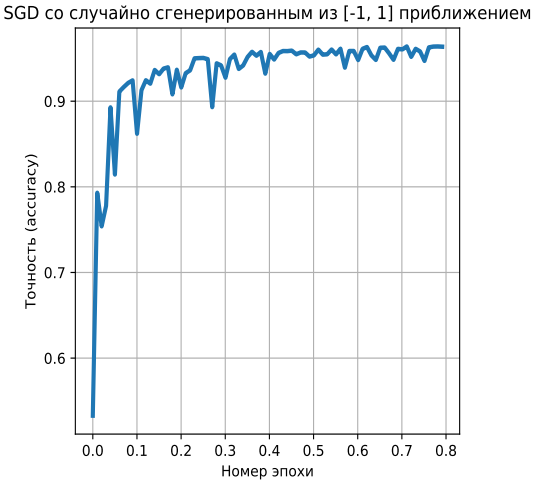

Рис. 8: Зависимость точности от приближенного номера эпохи стохастического градиентного спуска

Для измерения точности алгоритмов в зависимости от итерации (эпохи) и затраченного времени в методы fit классов $GDClassifier$ и $SGDClassifier$ был добавлен параметр $calc\_accuracy$, по умолчанию равный $False$. В случае $calc\_accuracy = True$ алгоритм обучался на части поданной на вход выборки (в данном эксперименте равной 70% от исходной), а точность вычислялась на остав-

шейся подвыборке. Графики зависимости точности от реального времени работы методов можно видеть на Рис. 5 и 6, а от номера итерации и эпохи — на Рис. 7 и 8.

Сравнивая приведенные графики с графиками на Рис. 1-4, можно заметить, что даже малому уменьшению функции потерь (на последних итерациях) соответствует небольшое увеличение точности. Это свидетельствует о том, что у модели, всё сильнее и сильнее подстраивающейся под обучающую выборку, обобщающая способность всё равно продолжает расти. Иными словами, при заданных параметрах модель не переобучается.

Исходя из вышесказанного, можно сделать вывод, что стохастический градиентный спуск является более подходящим методом для решения данной задачи.

Рассмотрим теперь модель, обученную с теми же параметрами с помощью стохастического градиентного спуска, но с другим начальным приближением. Полученные графики изображены на Рис. 9 и 10.

Рис. 9: Зависимость значения функции потерь от времени работы стохастического градиентного спуска

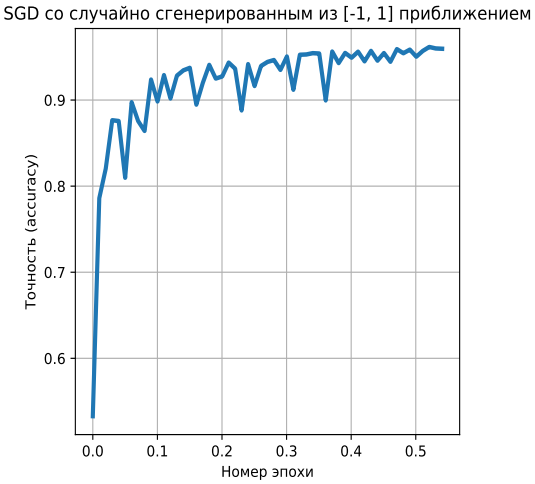

Рис. 10: Зависимость точности от номера эпохи стохастического градиентного спуска

Заметим, что по сравнению с предыдущим начальным приближением время работы алгоритма и количество эпох уменьшилось, но значение функции потерь и точность на отложенной выборки остались близки к ранее полученным результатам. Таким образом, в зависимости от начального приближения время работы метода может меняться, но финальные результаты, скорее всего, будут почти одинаковыми.

3 Исследование влияния параметров, задающих шаг градиентного спуска, на качество модели

Темп обучения (или же длина шага градиентного спуска) задается формулой $step = \alpha/k^{\beta}$, где k — номер итерации. Целью данного эксперимента было исследование влияния параметров α и β на качество работы методов.

Сперва были проверены значения α из множества $\{0.1,\ 1,\ 100,\ 1000\}$ при зафиксированном значении $\beta = 0$. Значения функции потерь и полученные точности отражены в Таб. 1 и на Рис. 11.

При значении $\alpha = 1000$ для полного и, в еще большей степени, для стохастического градиентного спуска были получены огромные значения функционала ошибок. Очевидно, что в связи с выбранным значением параметра алгоритм на каждой итерации делал слишком большой шаг, каждый раз «переступая» точку минимума (в случае полного), либо, возможно, уходя совсем не в том направлении (в случае стохастического).

	GD	SGD
$\alpha = 0.1$	0.4128	0.2028
$\alpha = 1$	0.2247	0.1359
$\alpha = 100$	0.2533	14.533
$\alpha = 1000$	29.082	204.38

Таблица 1: Значение функции потерь в зависимости от α

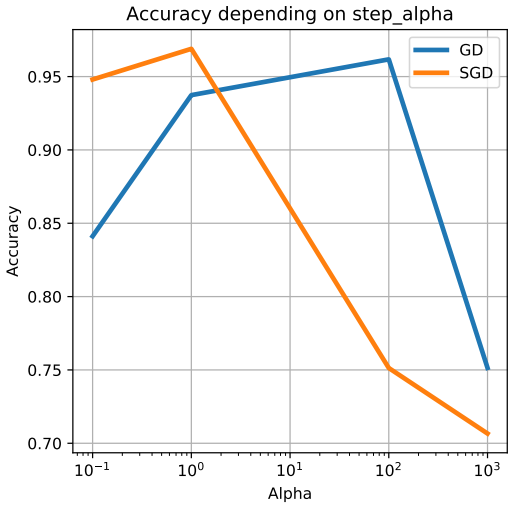

Рис. 11: Зависимость точности от α

При $\alpha = 100$ полный градиентный спуск показал хорошие результаты: для данного метода была достигнута наибольшая точность. В это же время стохастический градиентный спуск показал низкую точность и высокое значение функционала ошибок. Разницу в результатах можно объяснить тем, что в методе полного градиентного спуска на каждой итерации происходит гарантированное движение в сторону уменьшения функционала. Таким образом, в данном методе можно позволить себе делать «более уверенные» шаги. В методе же стохастического градиентного спуска движение, вообще говоря, не производится в сторону уменьшения функционала. Поэтому большие значения длины шага могут привести к нежелаемому результату.

При $\alpha = 1$ для обоих методов были достигнуты наименьшие значения функционала ошибок, а для стохастического градиентного спуска также и наилучшая точность. Таким образом, это значение параметра метода стохастического градиентного спуска является оптимальным для решения данной задачи.

При $\alpha = 0.1$ также были получены неплохие результаты. Но, сравнивая полученное значение функционала для полного градиентного спуска с соответствующими значениями при $\alpha = 1$ и $\alpha = 100$, можно сделать вывод, что длина шага была слишком мала, и алгоритм за отведенное количество итераций «не успел» максимально приблизится к точке оптимума.

Далее было зафиксировано значение $\beta = 0.5$ и проверены те же самые значения α. Полученные результаты отражены на Рис. 12 и 13.

Наилучшее качество моделей достигается на значениях $\alpha = 100$ и $\alpha = 1000$. Это связано с тем, что при выбранном значении β длина шага уменьшается с увеличением числа итераций, что обеспечивает сходимость метода и решает проблему «перешагивания» через точку минимума, которая имела место в предыдущем случае. В это

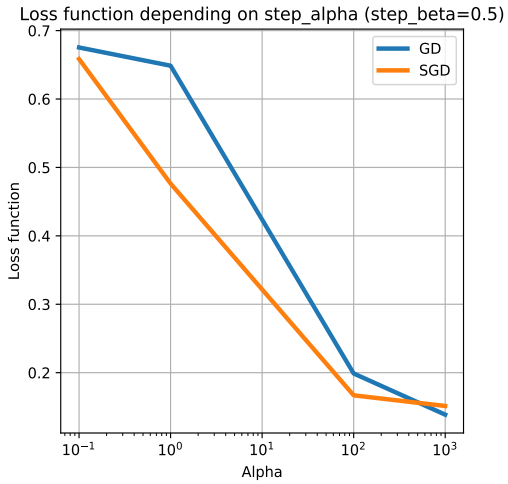

Рис. 12: Значение функции потерь в зависимости от α ($\beta = 0.5$)

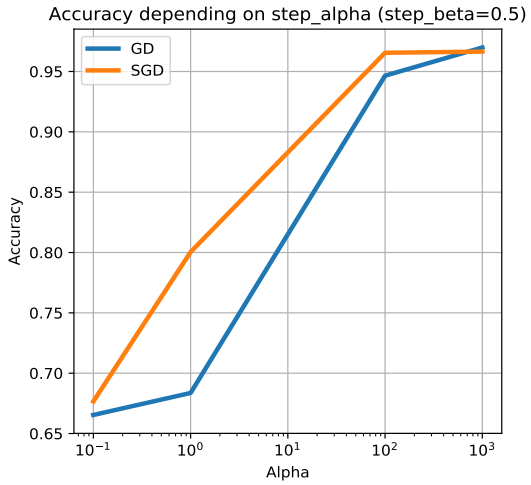

Рис. 13: Зависимость точности от α ($\beta = 0.5$)

же время малые значения α дали плохие результаты, поскольку с увеличением числа итераций длина шага становилась слишком малой, что не позволило алгоритму приблизиться к точке минимума.

4 Исследование влияния фактора случайности при выборе объектов на качество модели

В рамках этого эксперимента было обучено 5 моделей при помощи стохастического градиентного спуска с различными значениями параметра *random_seed*. В соответствии с результатами прошлого эксперимента для всех моделей были использованы параметры $\alpha = 1$ и $\beta = 0$. Полученные значения функции ошибок и точности моделей приведены в Таб. 2.

	$seed = 1$	$seed = 5$	$seed = 10$	$seed = 50$	$seed = 100$
Loss	0.1479	0.1369	0.1397	0.1408	0.1388
Accuracy	0.9572	0.9655	0.9676	0.9629	0.9678

Таблица 2: Значение функции потерь и точности в зависимости от *random_seed*

Исходя из полученных результатов, можно сделать вывод, что фактор случайности при выборе объектов не оказывает значительного влияния на качество модели.

5 Исследование влияния размера подвыборки на качество метода стохастического градиентного спуска

В данном эксперименте были рассмотрены значения размера подвыборки из множества {1, 5, 10, 20, 50, 100}. График зависимости функции потерь от времени работы метода стохастического градиентного спуска для моделей с различным

значением этого параметра изображен на Рис. 14, а полученные значения точности отображены в Таб. 3.

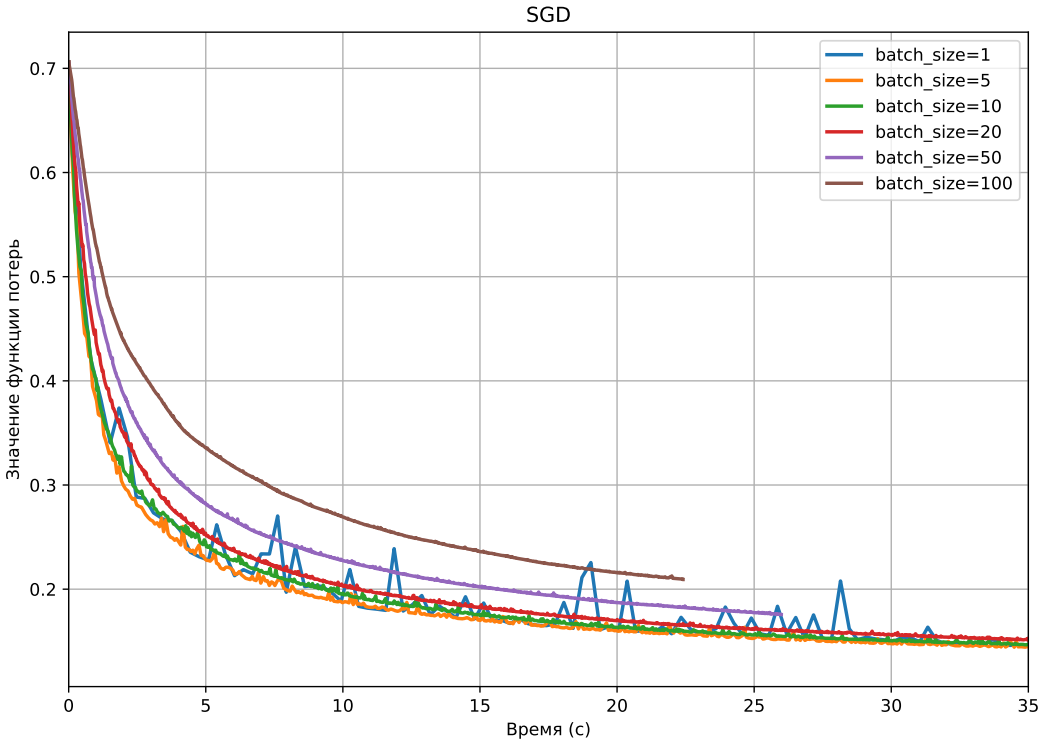

Рис. 14: Значение функции потерь в зависимости от времени работы стохастического градиентного спуска

	$batch = 1$	$batch = 5$	$batch = 10$	$batch = 20$	$batch = 50$	$batch = 100$
Accuracy	0.9684	0.9690	0.9691	0.9669	0.9564	0.9459

Таблица 3: Значение точности в зависимости от размера подвыборки

Из полученных результатов можно сделать некоторые выводы. Во-первых, с увеличением размера подвыборки модели чаще выходят по критерию остановки (модуль разности между значениями функций потерь на двух «соседних» эпохах меньше заданного параметра). Так, например, модели с $batch\_size = 50$ и $batch\_size = 100$ вышли из цикла «досрочно». Это связано с тем, что модели с большим размером подвыборки всё более приближаются к модели полного градиентного спуска, и их графики становятся более гладкими. Во-вторых, как и следовало ожидать, значения минимизируемого функционала для модели с $batch\_size = 1$ сильно отличаются на соседних замерах. Тем не менее, в конечном итоге данная модель показала высокую точность на отложенной выборке. В-третьих, модели с размером подбыворки 5, 10 и 20 показали похожие результаты, как с точки зрения минимизации функции потерь, так и с точки зрения полученной точности. Как видно из Таб. 3, наибольшее значение точности достигается на модели с $batch\_size = 10$.

Дополнительно было исследовано, как изменится зависимость точности модели от параметра α при значении $batch\_size = 10$. Сравнивая полученный график, приведенный на Рис. 14, с аналогичными результатами для модели с $batch\_size = 1$

(Рис. 11), можно заметить, что точность значительно увеличилась при значении $\alpha = 100$. Данный факт может быть объяснен тем, что с увеличением размера подвыборки в методе стохастического градиентного спуска направление движения с бóльшей вероятностью совпадет с направлением уменьшения функционала.

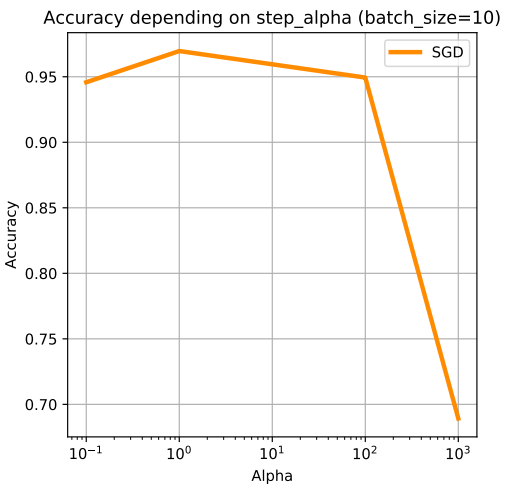

Рис. 15: Зависимость точности от α ($batch\_size = 10$)

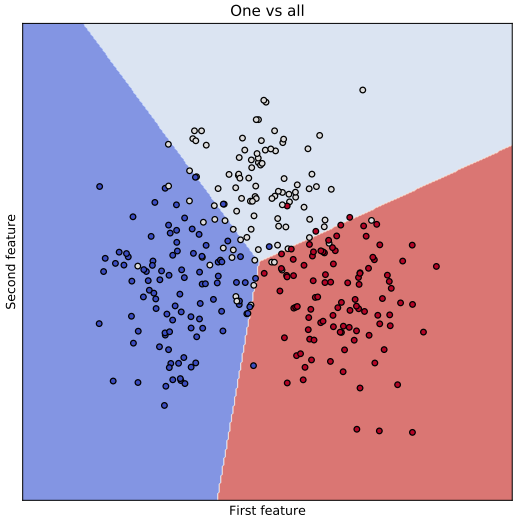

Рис. 16: Метод один против всех

6 Исследование особенностей различных стратегий решения многоклассовых задач классификаций

В данном эксперименте были исследованы особенности трех разных способов решения задачи многоклассовой классификации: один против всех, каждый против каждого и многоклассовая логистическая регрессия.

Эксперимент проводился на модельных данных с 2 признаками и 3 классами, которые были предварительно отмасштабированы.

Решения данной задачи классификации вышеперечисленными методами изображены на Рис. 16-18.

Все три метода достаточно хорошо справились с поставленной задачей. Наиболее равномерно признаковое пространство было разбито разделяющими плоскостями метода каждый против каждого (Рис. 17). Этот же метод показал наибольшую точность. Разделяющие плоскости метода один против всех и мультиномиальной логистической регрессии довольно близки друг к другу. Данные методы показали примерно одинаковые значения точности классификации.

Несомненными недостатками методов один против всех и каждый против каждого являются затрачиваемые ресурсы. В первом случае обучается K бинарных классификаторов, а во втором — C_K^2, что влечет за собой большие затраты времени при обучении. Мультиномиальная регрессия такой проблемой не обладает, поскольку все K векторов весов обучаются одновременно. Таким образом, если вычислительные и временные ресурсы сильно ограничены, мультиномиальная регрессия является наиболее подходящим инструментом для решения подобной задачи.

С другой стороны, если, например, время обучения модели не столь важно (допустим, необходимо обучить модель лишь единожды, а далее только делать пред-

сказания), то стратегии каждый против каждого и один против всех могут быть неплохой альтернативой мультиномиальной регрессии и даже, возможно, позволят получить лучшие результаты (как, например, в данном эксперименте).

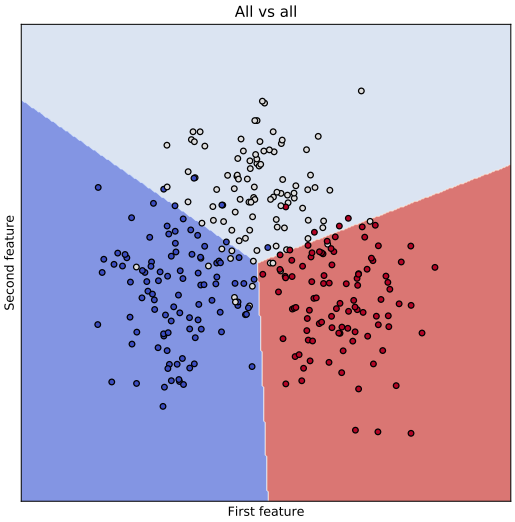

Рис. 17: Метод каждый против каждого

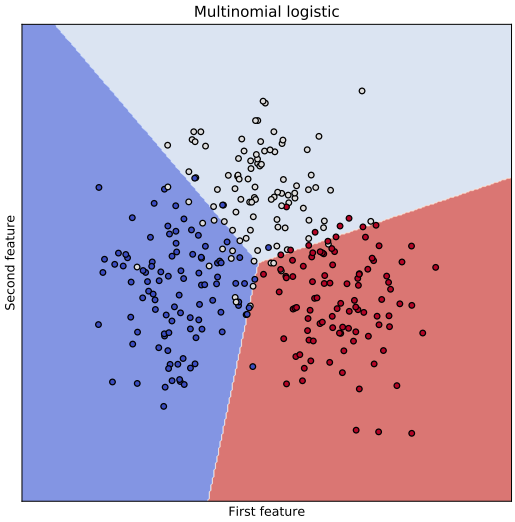

Рис. 18: Мультиномиальная регрессия

7 Исследование влияния различных параметров модели мультиномиальной регрессии на качество классификации датасета 20newsgroups

В данном и всех последующих экспериментах исследования проводились на наборе данных 20newsgroups. Предварительно во всех документах были убраны заголовки, цитаты и подписи, а также все буквы были переведены нижний регистр. Также все символы, не являющиеся буквами, цифрами и апострофом были заменены на пробелы. Решение не заменять апострофы было обусловлено нежеланием разбивать слова наподобие «don't» или «I'm» на два. Далее датасет был преобразован в разреженную матрицу, где значение x в позиции (i, j) означает, что в документе i слово j встретилось x раз. После этого было произведено tf-idf преобразование датасета.

7.1 Подбор параметра α

Напомним, что длина шага градиентного спуска задается формулой $step = \alpha/k^{\beta}$, где k — номер итерации. Подбор параметров было решено начать с выбора α при зафиксированных $batch\_size = 3$ и $max\_iter = 2000$, а также $\beta = 0$. Были проверены значения параметра $\alpha \in \{0.1, 1, 5, 10, 25, 50, 100\}$. Полученные графики приведены на Рис. 19 и 20

Из Рис. 19 можно заметить, что модели с маленьким значением данного параметра ($\alpha = 0.1$ и $\alpha = 1$) не достигли желаемого значения функции потерь, поскольку двигались к точке минимума слишком медленно.

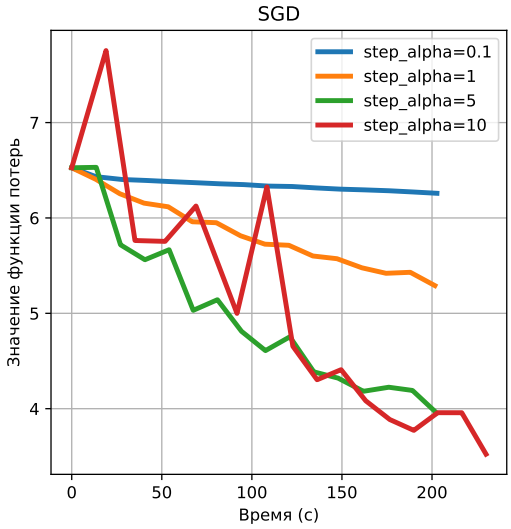

Рис. 19: Значения функций потерь в зависимости от времени

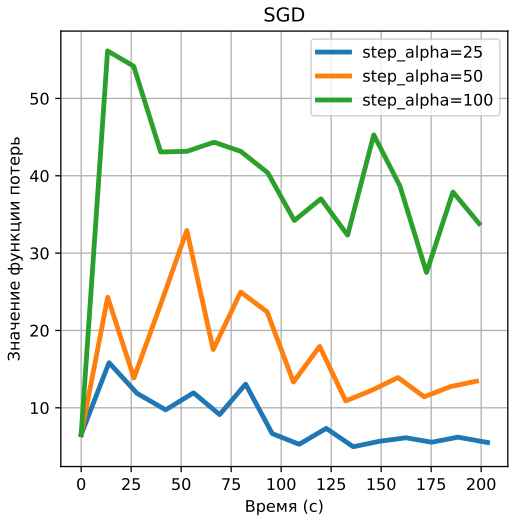

Рис. 20: Значения функций потерь в зависимости от времени

Из Рис. 20 прослеживается медленная сходимость (либо, возможно, ее отсутствие) моделей с $\alpha = 50$ и $\alpha = 100$. У данных моделей значение функции потерь, полученное на выходе алгоритма, даже превышает стартовое.

Хорошие результаты по скорости сходимости и финальному значению минимизируемого функционала показали модели с $\alpha = 5$, $\alpha = 10$ и $\alpha = 25$. Для проведения дальнейших исследований было выбрано значение параметра $\alpha = 10$, поскольку именно этой моделью было достигнуто наименьшее значение функционала.

7.2 Подбор параметра *batch_size*

Графики функций потерь моделей стохастического градиентного спуска с различными значениями размера подвыборки приведены на Рис. 21.

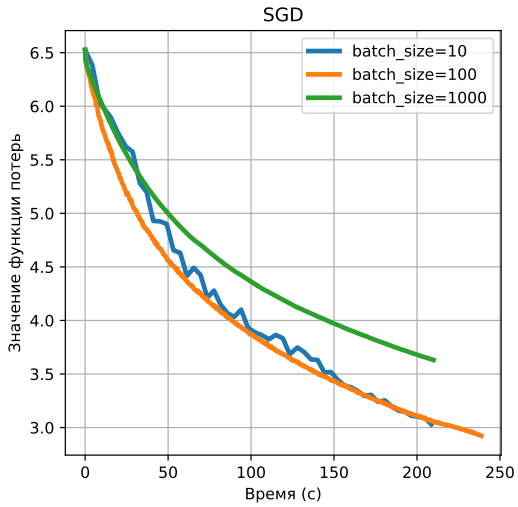

Рис. 21: Значения функций потерь в зависимости от времени

Из приведенного графика видно, что за одинаковый промежуток времени модель с $batch\_size = 100$ достигает меньшего значения функционала по сравнению с другими моделями. Этот факт послужил основанием для выбора данного значения параметра для проведения последующих экспериментов.

7.3 Подбор параметра max_iter

Для нахождения оптимального количества итераций были обучены две модели со значениями параметра $max\_iter = 10\,000$ и $max\_iter = 15\,000$. Полученные графики зависимости функции ошибок от времени работы методов приведены на Рис. 22 и 23.

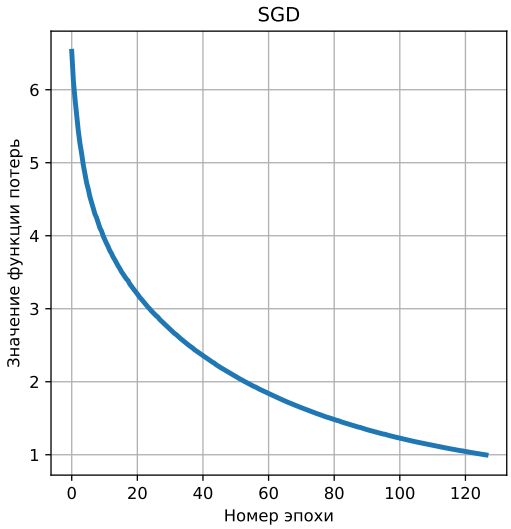

Рис. 22: Значение функции потерь в зависимости от номера эпохи ($max\_iter = 10\,000$)

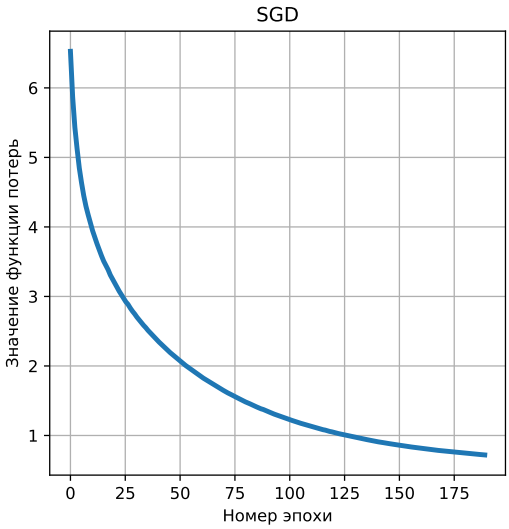

Рис. 23: Значение функции потерь в зависимости от номера эпохи ($max\_iter = 15\,000$)

Заметим, что на графике справа функция потерь между 120-ой и 180-ой эпохой продолжает уменьшаться, но крайне незначительно. Такая минимизация функционала не оправдана с точки зрения затрачиваемого времени.

Более того, для данных моделей была подсчитана точность на отложенной выборке. Модель, обученная на 15 000 итераций, показала точность 72.7%, а модель, обученная на 10 000 итераций, — 73.3%!

Таким образом можно сделать вывод, что вторая модель начала слишком сильно подгоняться под обучающую выборку, что повлекло за собой уменьшение обобщающей способности алгоритма. Исходя из всего вышесказанного, для дальнейших исследований было выбрано значение параметра $max\_iter = 10\,000$.

7.4 Подбор параметра β

В данном эксперименте были проверены значения параметра $\beta \in \{0.001, 0.01, 0.1, 0.5\}$. Полученные результаты приведены на Рис. 24.

При значении $\beta = 0.5$ длина шага с увеличением номера итерации уменьшалась слишком быстро, что не позволило методу приблизиться к точке минимума функции потерь. Результаты моделей с $\beta = 0.01$ и $\beta = 0.001$ крайне близки к полученным ранее

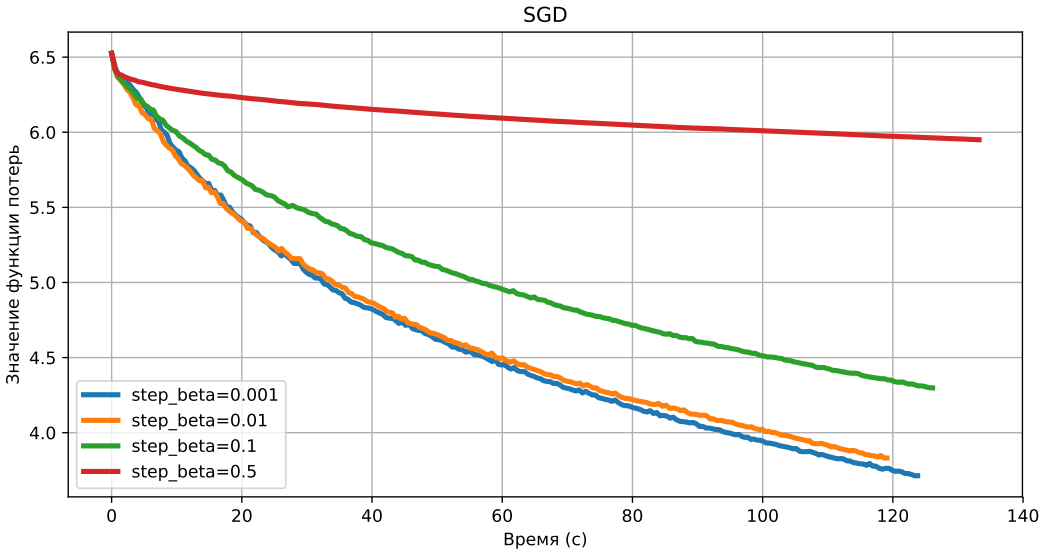

Рис. 24: Значения функций потерь в зависимости от времени

результатам для модели $\beta = 0$. Тем не менее, из общих соображений было решено выбрать значение параметра $\beta = 0.01$, поскольку небольшое уменьшение длины шага по мере увеличения количества итераций кажется довольно логичным действием, а также соответствует идеи метода градиентного спуска.

7.5 Итоговые значения параметров

Во время проведения предыдущих экспериментов эмпирическим путем был подобран параметр $tolerance = 10^{-6}$. При бóльших значениях параметра метод зачастую выходил «досрочно», не достигая желаемого значения функции. При меньших значениях возрастало затрачиваемое методом время.

Таким же образом был подобран коэффициент регуляризации $\lambda = 10^{-5}$. При данном значении параметра в ходе экспериментов на отложенной выборке была достигнута довольно высокая точность. При бóльших же значениях модель становилась неустойчивой (на начальных итерациях значения функции потерь сильно возрастали), а при меньших значениях присутствует бóльшая вероятность переобучения.

Суммируя полученные результаты, финальные параметры модели выглядят следующим образом:

$$\alpha = 10,\ batch\_size = 100,\ max\_iter = 10\,000,\ tolerance = 10^{-6},\ \beta = 0.01,\ \lambda = 10^{-5}$$

В завершении этой группы экспериментов была обучена модель с вышеуказанными параметрами. Точность на отложенной выборке составила 73.4%, а продолжительность процесса обучения - 17 минут 36 секунд.

7.6 Применение лучшей модели к данным, не преобразованным с помощью tf-idf

С целью исследования влияния tf-idf преобразования на точность, модель с вышеперечисленными параметрами была обучена на датасете, который соответствует

описанию в начале пункта 7, но без применения tf-idf преобразования. График функции потерь в зависимости от количества эпох приведен на Рис. 25.

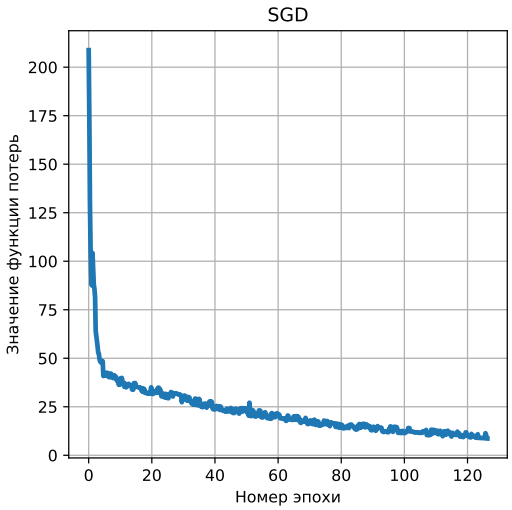

Рис. 25: Значения функций потерь в зависимости от номера эпохи

Необходимо отметить, что на таких данных начальное приближение дает огромное значение функционала ошибок и, следовательно, для его минимизации требуется больше времени. В результате была получена точность, равная 59%, что намного меньше достигнутой точности на tf-idf преобразованном датасете.

8 Сравнение точности на тестовой выборке с точностью на отложенной и анализ ошибок модели

Тестовая выборка предварительно была подвергнута точно таким же преобразованиям, что и обучающая ранее. Применив выбранную модель к тестовой выборке, была достигнута точность 67.7%, что примерно на 5% ниже точности, полученной на отложенной выборке. Данный результат был ожидаем, поскольку результаты на новых тестовых данных обычно хуже результатов, полученных при обучении. Также метод проверки точности на отложенной выборке имеет свои недостатки. Например, при различных разбиениях исходной выборки на отложенную и обучающую получающиеся значения точности могут довольно сильно отличаться.

Приведем несколько примеров ошибочно классифицированных текстов:

- «i'm not familiar at all with the format of these x face thingies but after seeing them in some folks' headers i've got to see them and maybe make one of my own i've got dpg view on my linux box which displays uncompressed x faces and i've managed to compile un compface too but now that i'm looking for them i can't seem to find any x face 's in anyones news headers could you would you please send me your x face header i know i'll probably get a little swamped but i can handle it i hope»

  Данный текст принадлежит к классу «comp.windows.x», алгоритм же отнёс его к «comp.graphics».

- «fine but one of the points of this entire discussion is that we conservative reformed christians this could start an argument but isn't this idea that homosexuality is ok fairly new this century is there any support for this being a viable viewpoint before this century i don't know don't believe that homosexuality is acceptable to him so your scripture quotation doesn't work for us»

  Истинный класс данного текста — «soc.religion.christian», алгоритм предсказал «alt.atheism».

- «i have this kit which includes the following 1 82c84a 82c84a 5 chmos clock generator and driver for 8086 80c88 processors 2 27c64 87c64 64k 8kx8 chmos uv erasable prom 3 51c259l low power 64k x 4 chmos dynamic ram 4 82c59a 2 chmos programmable interrupt controller 5 82c88 chmos bus controller fro 80c86 80c88 processors 6 80c88 80c88 2 8 bit chmos microprocessor 7 82c55a chmos programmable peripheral interface 8 82c54 chmos programmable interval timer 9 82c08 chmos dynamic ram controller all these are chips with complete manual in a box i don't know whether they still work or not and i don't really know what they are so this is mainly for those who knows what this is and have use of it probably ee stuff since this used to belong to a ee student anyone interested please make me an offer»

  Данный текст принадлежит к классу «misc.forsale», а алгоритм отнёс его к «sci.electronics».

В первых двух примерах истинная и предсказанная темы текстов достаточно близки по своему смысловому значению. В третьем примере кажется, что темы «forsale» и «electronics» ни коим образом не связаны, но прочитав данный текст, можно понять, что речь идет о продаже электронной техники.

Таким образом, рассмотренные примеры, на которых алгоритм допускал ошибку, были действительно непросты для классификации, поскольку могли быть отнесены к нескольким темам.

Матрица ошибок модели изображена на Рис. 26.

Проанализировав матрицу ошибок, можно сделать следующие замечания:

- Алгоритм 132 раза правильно классифицировал объекты из темы 18, что соответствует «talk.politics.misc», но в это же время целых 100 раз ошибочно отнес такие объекты к теме 16 («talk.politics.guns»). Такое количество ошибок связано со смысловой схожестью данных тем

- Предсказав правильно для 88-ти объектов тему 19 («talk.religion.misc»), алгоритм 46 раз ошибочно относил объекты этого класса к теме под меткой 0 («'alt.atheism'») и 41 раз к теме под меткой 15 («soc.religion.christian»). Все три темы связаны с религией, а значит, скорее всего, в них используются похожие термины, чем и объясняются ошибки модели.

Таким образом, большинство ошибочно классифицированных объектов были отнесены к темам, имеющим близкое к истинному смысловое значение.

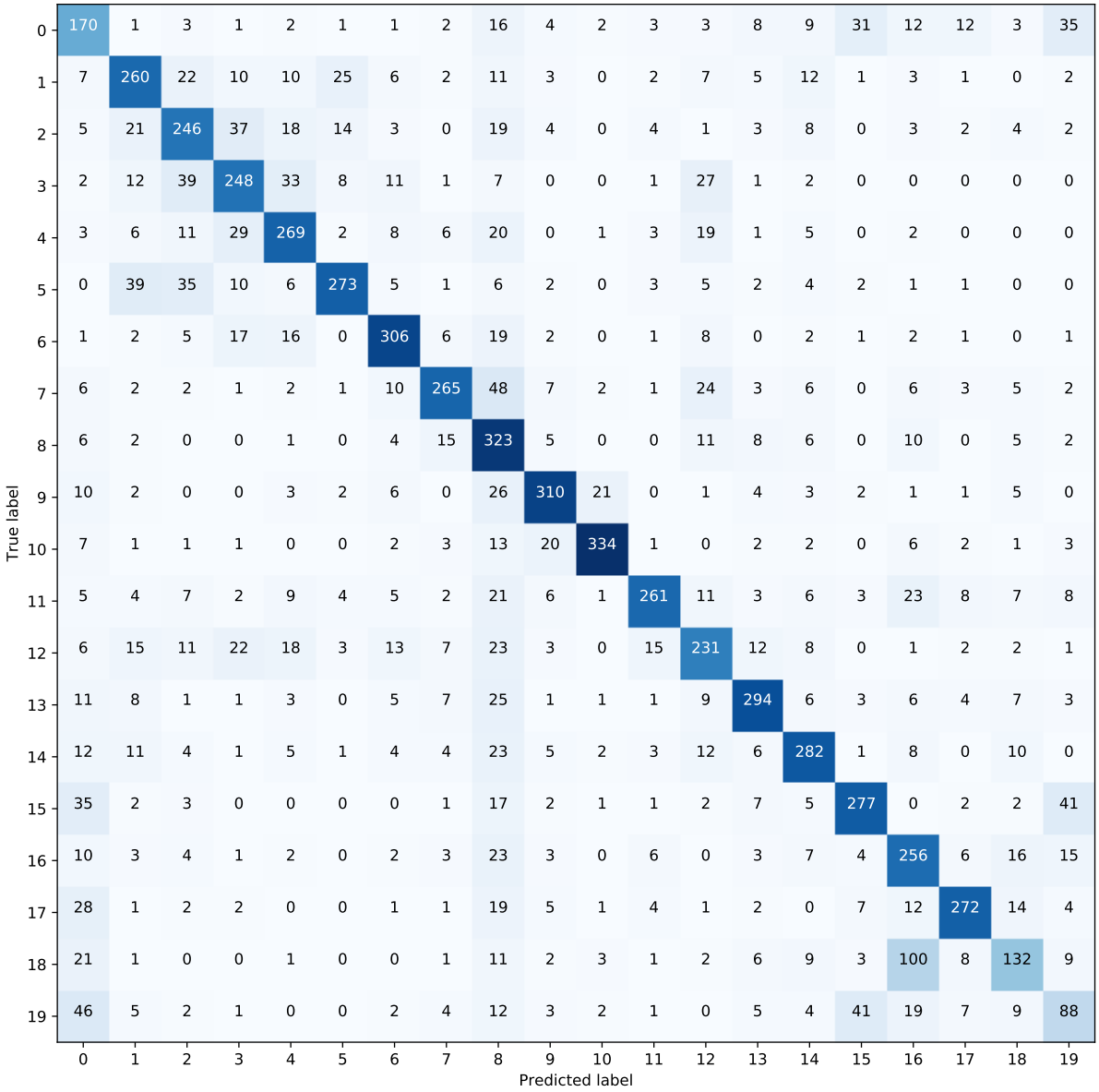

Рис. 26: Матрица ошибок

9 Исследование влияния предобработки текста на качество модели

9.1 Применение стемминга

В данном эксперименте тексты предварительно были подвергнуты процедуре стемминга, в результате чего количество признаков уменьшилось примерно на 17 000 (с 98 698 до 81 570). Применив модель с подобранными ранее параметрами, была достигнута точность 68.2%, что является улучшением результата модели на непредобработанных данных на 0.5%.

Время работы алгоритма составило 14 минут 25 секунд, что быстрее предыдущего результата на 3 минуты 11 секунд. Уменьшение времени работы модели прямым

образом связано с уменьшением размерности признакового пространства.

9.2 Применение лемматизации

После применения лемматизации количество признаков уменьшилось примерно на 8000 (с 98 698 до 90 999). В этот раз обучение модели составило 16 минут 23 секунд, что быстрее модели на непредобработанных данных примерно на минуту. Была достигнута точность 67.93%. Полученная точность меньше точности, достигнутой после стемминга, но всё ещё превосходит точность, полученную на непредобработанных данных.

Таким образом, для данной задачи как стемминг, так и лемматизация сократили размерность признакового пространства, уменьшили время работы модели и увеличили точность её предсказания.

10 Улучшение качества алгоритма за счёт сокращения словаря

Поскольку стемминг показал себя наилучшим образом в предыдущем эксперименте, сокращению подвергнется словарь, полученный именно после такой предобработки текстов.

Сначала сокращение словаря будет проводится за счёт выбрасывания самых часто встречающихся слов. Полученные значения точности приведены в Таб.4

	$n = 10$	$n = 25$	$n = 50$	$n = 75$	$n = 100$	$n = 250$
Accuracy	0.6779	0.6798	0.6854	0.6836	0.6848	0.6771

Таблица 4: Значение точности в зависимости от количества выброшенных частотных слов n

Наибольшая точность была достигнута при 50 выброшенных частотных слов. С помощью сокращения словаря таким способом удалось увеличить значение точности на 0.34%. Время работы алгоритма по сравнению с предыдущим экспериментом практически не изменилось (уменьшение размерности признакового пространства незначительно).

Повторим предыдущий эксперимент, но теперь сокращение словаря будет обеспечиваться за счет выбрасывания самых редких слов. Полученные результаты приведены в Таб.5

	$n = 10$	$n = 25$	$n = 50$	$n = 75$	$n = 100$	$n = 250$
Accuracy	0.6796	0.6788	0.6798	0.6801	0.6779	0.6765

Таблица 5: Значение точности в зависимости от количества выброшенных редких слов n

Как видно из полученных результатов, выбрасывание редких слов не привело к улучшению качества модели.

В заключение было исследовано, как изменяется качество алгоритма, если сокращать словарь за счет выбрасывания стоп-слов. Точность модели, обученной на

полученном датасете, составила 68.38%, что уступает значению точности, полученной после удаления 50 самых частотных слов, но тем не менее является улучшением результата, полученном после применения стемминга без сокращения словаря. Время работы модели также практически не изменилось.

11 Заключение

Таким образом, в ходе выполнения практического задания «Градиентные методы обучения линейных моделей» была написана собственная реализация линейного классификатора с логистической функцией потерь (а также ее обобщение на случай многоклассовой классификации), реализованы общие стратегии многоклассовой классификации на основе бинарного классификатора один против всех и каждый против каждого. Также после осуществления подбора параметров линейной модели, предобработки данных с помощью стемминга и сокращения словаря за счёт выбрасывания частотных слов на датасете 20newsgroup удалось достигнуть точности предсказания 68.54%.

Список литературы

[1] LIBSVM Data: Classification

[2] 20 Newsgroups

[3] Визуализация матрицы ошибок

[4] Воронцов К. В., LaTeXв примерах, 2005