Московский государственный университет
имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики

Отчёт по заданию № 2

Выполнил:
студент 317 группы
Находнов М. С.

Москва
2018

Содержание

Введение

В данном отчёте описаны эксперименты, направленные на изучение качества и скорости работы линейных алгоритмов классификации, а именно бинарной классификации (логистическая регрессия) и различных видов многоклассовой классификации (мультиномиальная классификация, подходы один-против-всех и каждый-против-каждого)

Все эксперименты проведены на отзывах из датасета 20newsgroups (- размер обучающей выборки, - размер тестовой выборки).

Эксперименты

Логистическая регрессия

В данном разделе будут рассмотрены эксперименты с подвыборкой из исходного датасета, а именно положительные и отрицательные отзывы. Будут рассмотренны методы градиентного спуска (GD) и стохастического градиентного спуска (SDG). Также будет произведёт анализ влияния гиперпараметров на точность и скорость сходимости этих методов.

Теоритические обоснования

Для реализации GD и SGD необходимо определить направление наискорейшего убывания функции потерь. В данном методе функция потерь имеет вид:

$$\mathbb{L} = -\frac{1}{N}\sum_{i=1}^{N}\ln(1+\exp(-y_i x_i w)) + \frac{\lambda}{2}||w||_2^2$$

Где $w \in \mathbb{R}^d$ - вектор столбец параметров, $x \in \mathbb{R}^{N*d}$ - множество объектов признаков, $y \in \{-1,1\}^N$ - множество меток классов. Несложно убедиться, что:

$$\nabla\mathbb{L}_w = \frac{1}{N}\sum_{i=1}^{N} y_i x_i \frac{\exp(-y_i x_i w)}{1+\exp(-y_i x_i w)} + \lambda w$$

Градиентный спуск

В качестве гиперпараметров данного алгоритма рассматриваются параметры определяющие величину шага $lr = \frac{\alpha}{\beta^{n_{iter}}}$, где n_{iter} - номер текущей эпохи. Были произведены замеры функции потерь и точности на обучающей и тестовой выборках в зависимости от номера итерации при различных значениях параметров α, β:

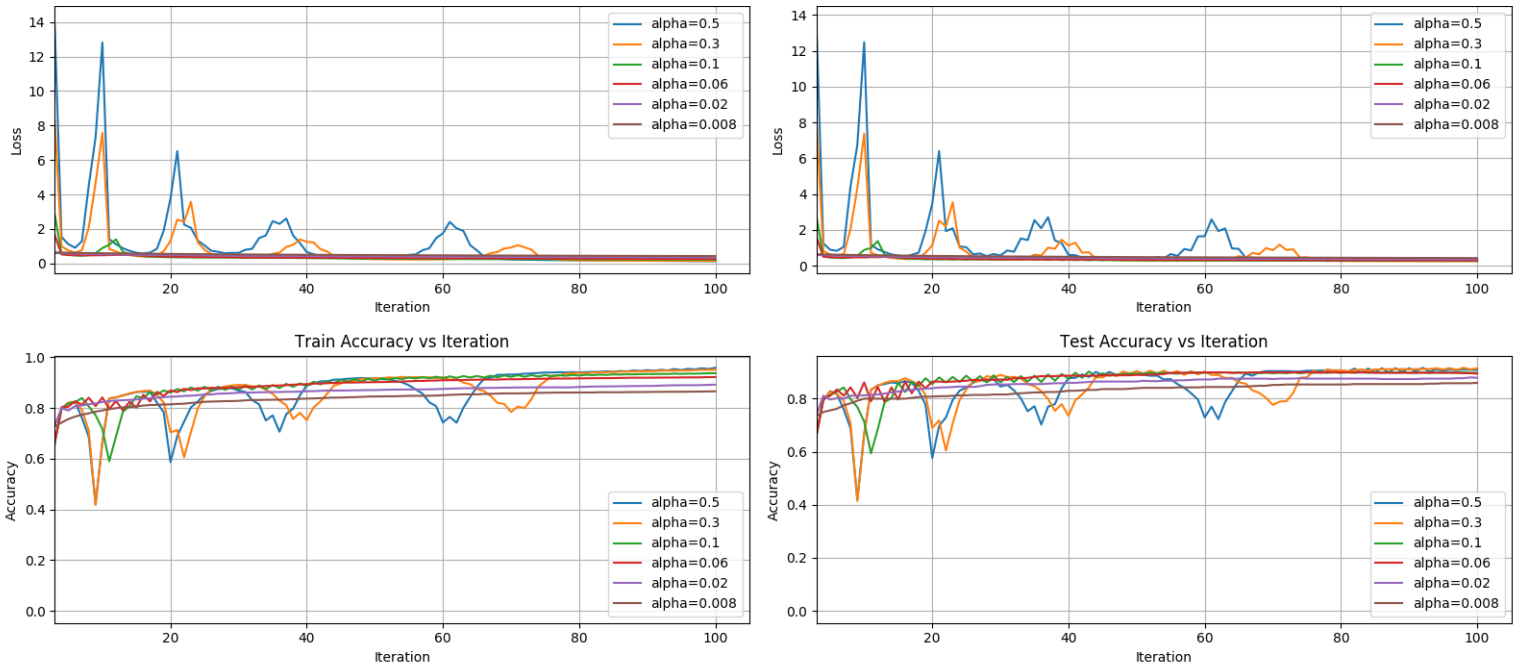

Рис. 1: Точность и функция потерь в зависимости от числа итераций при различных значениях α. (При $\beta = 0$)

Из 1 можно видеть, что с уменьшением α сходимость метода на начальных итерациях улучшается и графики функции потерь и точности имеют меньшую дисперсию. В тоже время, с увеличением числа итераций данный эффект нивелируется за счёт стремления нормы градиента к 0. Однако, более высокие значения α показывают лучшие результаты при большом числе эпох.

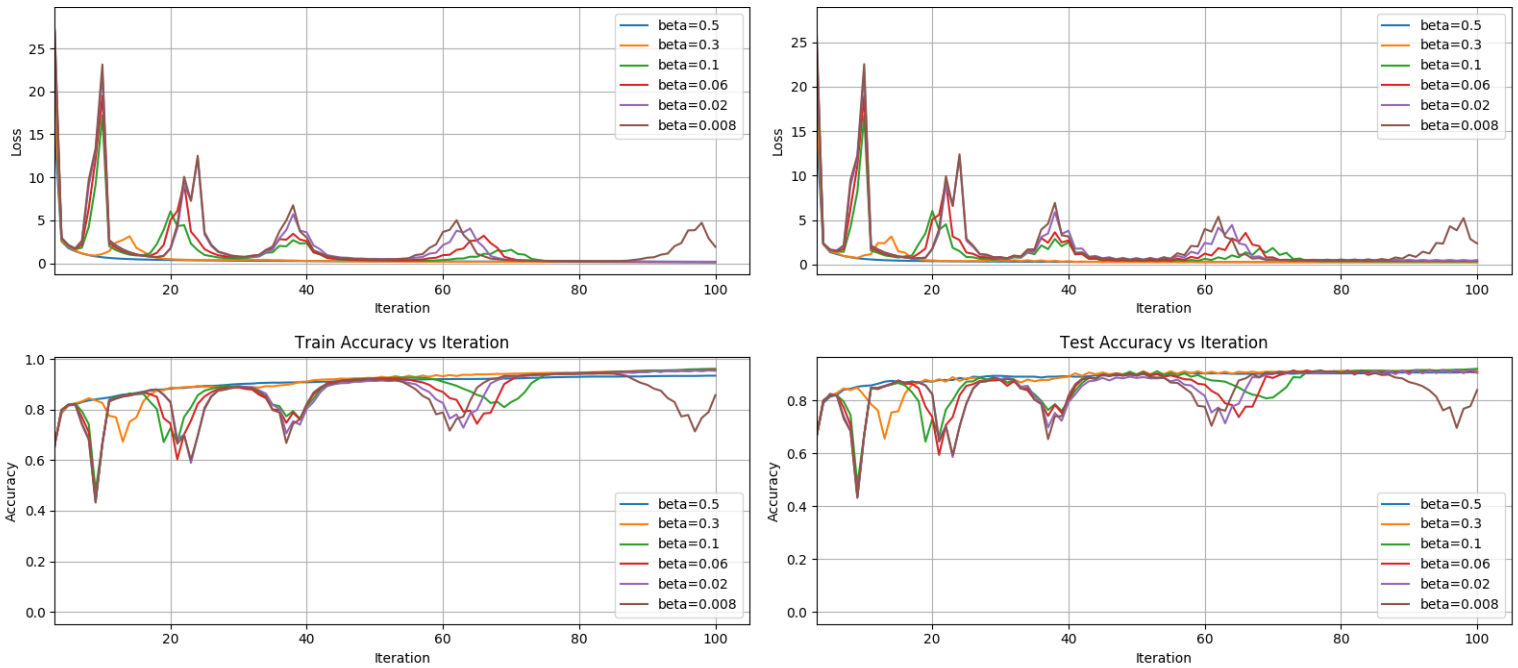

Рис. 2: Точность и функция потерь в зависимости от числа итераций при различных значениях β. (При $\alpha = 1$)

Как можно видеть на рисунке 2 поведение графиков в зависимости от β схоже с зависимостью от α - большие значения β приводят к значительной амплитуде колебаний

на начальных итерациях, но с ростом их числа, функция потерь и точность не только достигают тех же значений, что при малых β, но и незначительно их превосходят.

Как известно, стохастический градиентный спуск позволяет не только увеличить скорость работы алгоритмов, но и привести к лучшим результатам как относительно точности, так и функции потерь. Сравним градиентный спуск, описанный в предыдущем разделе со стохастическим градиентным спускам в зависимости от тех же гиперпараметров. Так же исследуем реальное время работы алгоритма и его точность в зависимости от различных значений размера батча.

Стохастический градиентный спуск

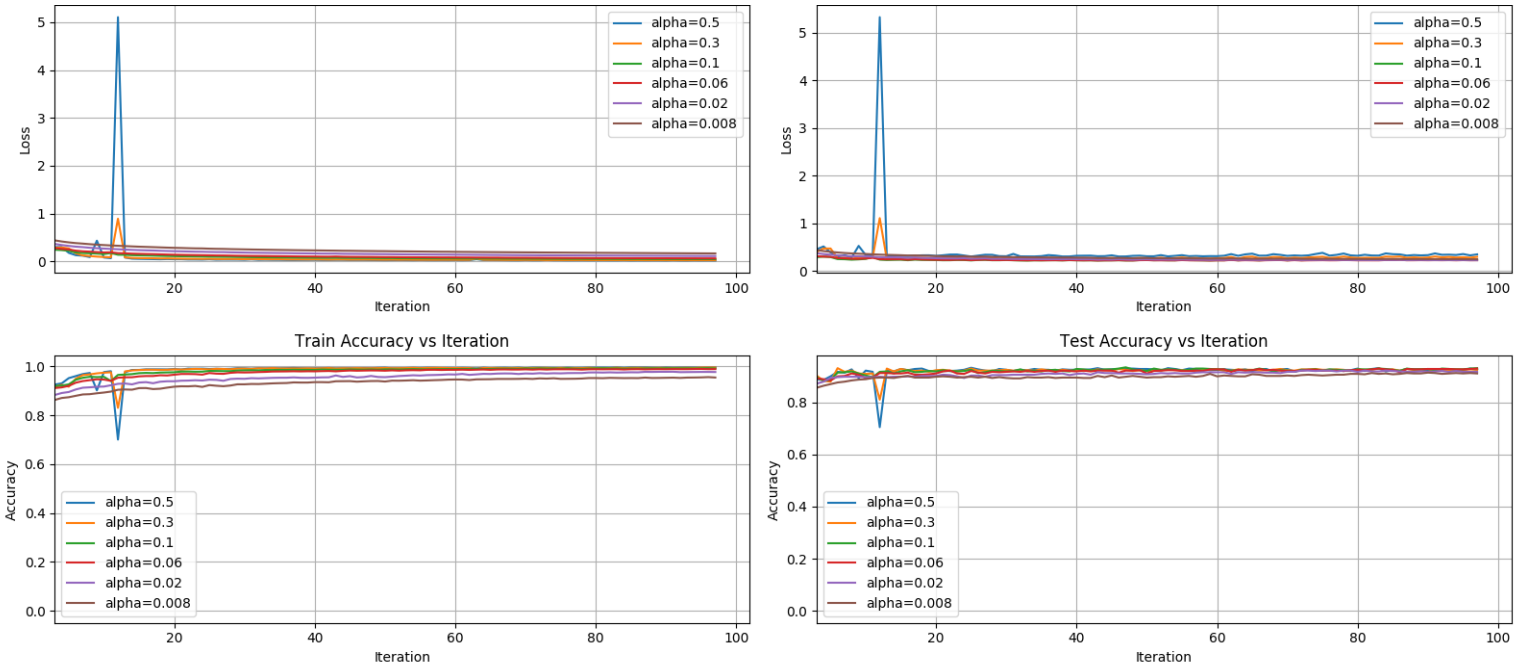

Рис. 3: Точность и функция потерь в зависимости от числа итераций при различных значениях α. (При $\beta = 0$)

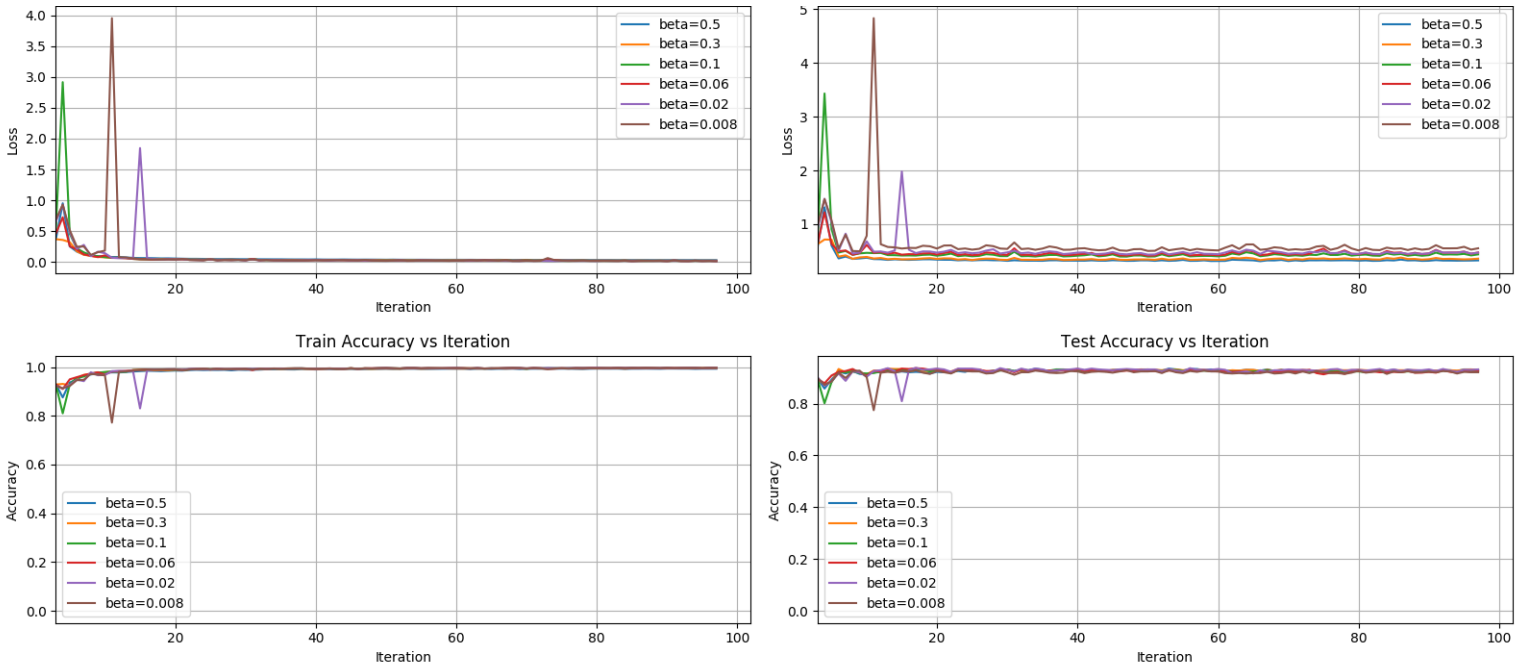

Рис. 4: Точность и функция потерь в зависимости от числа итераций при различных значениях β. (При $\alpha = 1$)

Как можно видеть из рисунков 3, 4, стохастический градиентый спуск позволяет методу более стабильно сходится в широком диапазоне значений α, β. Таким образом, SGD намного проще обучать за счёт более простого подбора гиперпараметров. Сравним точность и значения функции потерь для GD и SGD при одних и тех же значениях гиперпараметров:

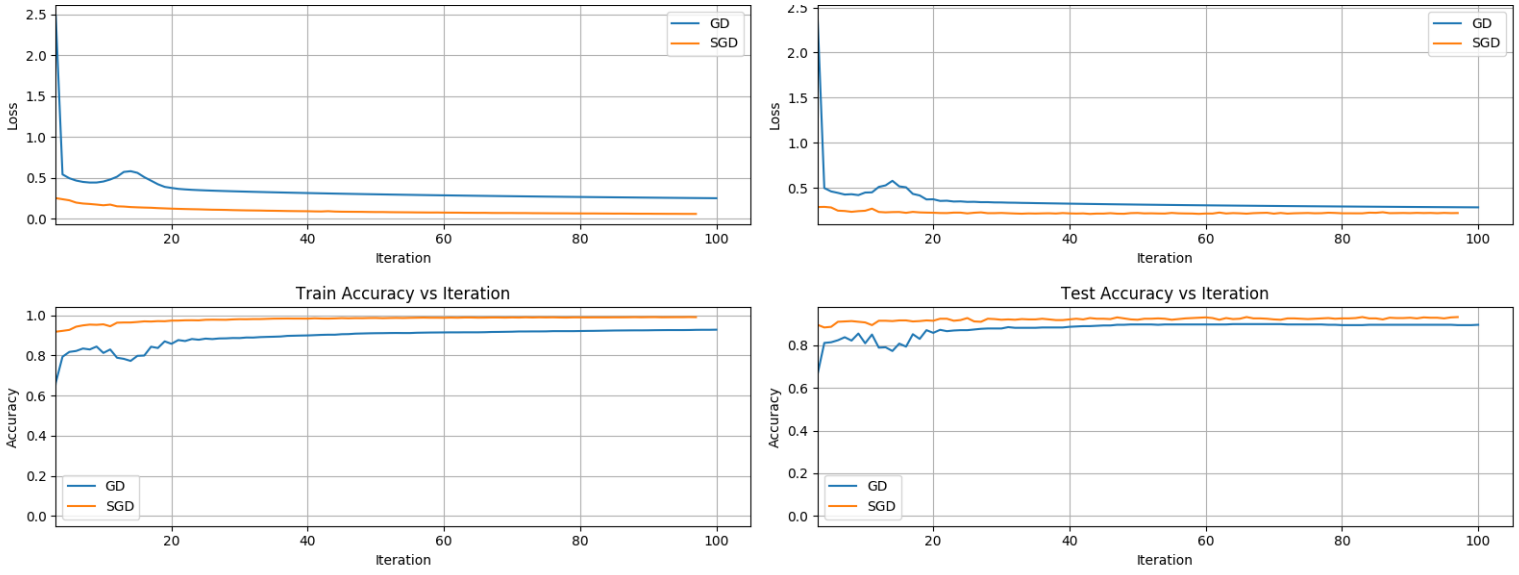

Рис. 5: Точность и функция потерь в зависимости от числа итераций для GD и SGD

Можно видеть, что SGD превосходит GD как по точности, так и по значениям функции потерь.

В конце, была исследована зависимость работы алгоритма в зависимости от размера батча:

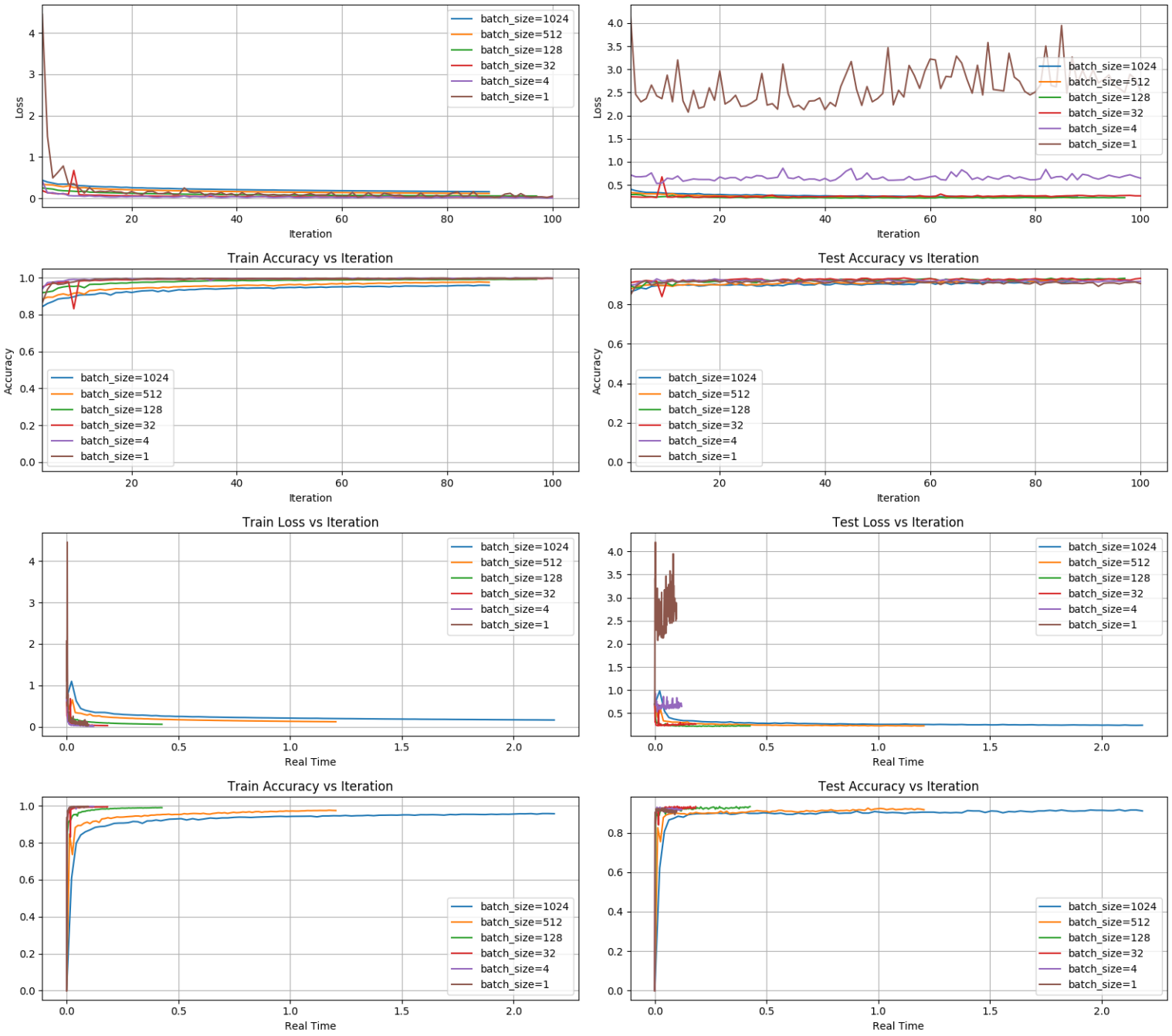

Рис. 6: Точность и функция потерь в зависимости от числа итераций при различных размерах батча ($\alpha = 0.1, \beta = 0.1$)

Из рисунка 6 видно несколько зависимостей:

1. Чем больше размер батча, тем дольше времени требуется, чтобы сделать одно и тоже число итераций

2. Слишком маленькие размеры батча обучаются намного хуже, чем умеренные или большие размеры

3. Начиная с размера батча 32 нет существенной разницы в качестве работы алгоритмов.

Выводы

В качестве результата, можно выделить, что SDG сходится стабильнее, быстрее и, в пределе, к лучшему результату нежели SD.

Многоклассовая классификация

В данном разделе будут проведены эксперименты с исходным датасетом. Будут проведено сравнение мультиномиальной регрессии и методов один-против-всех, каждый-против-каждого. Также будет произведёт анализ влияния гиперпараметров на точность и скорость сходимости этих методов.

Теоритические обоснования

Аналогично предыдущему пункту запишем функцию потерь в тех же обозначениях, за исключением того, что $w \in \mathbb{R}^{d*C}$ - матрица признаков:

$$\mathbb{L} = -\frac{1}{N}\sum_{i=1}^{N} x_i w_{y_i} + \frac{1}{N}\sum_{i=1}^{N} \ln(\sum_{k=1}^{C} \exp(x_i w_k)) + \frac{\lambda}{2}||w||_2^2$$

Дифференцируя функцию потерь поэлементно, получим:

$$\frac{\partial \mathbb{L}}{\partial w_{jp}} = -\frac{I}{N}\sum_{i=1}^{N} x_{ip}[\mathbb{I}[j = y_i] - \mathbb{P}(y_i = j|x_i)] + \lambda w_{jp}$$

$$\mathbb{P}(y_i = j|x_i) = \frac{\exp(x_i w^j)}{\sum_{k=1}^{C} \exp(x_i w^k)}$$

Подставляя $C = 2$ несложно убедиться, что данная формула переходит в аналогичную для логистической регрессии.

Выбор оптимального алгоритма

В данной секции будет произведено сравнение трёх вышеназванных методов с целью выбора оптимального для дальнейших экспериментов. Для методов один-против-всех и каждый-против-каждого в качестве базового алгоритма используется SGD с параметрами $\alpha = 0.1, \beta = 0.1$ с максимальным числом итераций 100 и размером батча 128. Для мультиномиальной регрессии так же используется SGD с теми же гиперпараметрами, но число итераций ограничено 10 ввиду значительно более медленной скорости работы алгоритма. Так же, экспериментальным путём обнаружено, что MN работает лучше всего при размере батча 1, поэтому именно он будет использоваться в дальнейшем.

-	One-vs-All	All-vs-All	MN
Accuracy (Train/Test)	0.722/0.665	0.652/0.606	0.666/0.621

Таблица 1: Точность для различных алгоритмов мультиклассовой классификации

Для улучшения работы алгоритма была применена лемматизация (пакет pymystem) и удаление стоп-слов.

-	One-vs-All	All-vs-All	MN
Accuracy (Train/Test)	0.722/0.665	0.652/0.606	0.772/0.697

Таблица 2: Точность для различных алгоритмов мультиклассовой классификации после лемматизации и удаления стоп-слов

Несмотря на то, что качество увеличилось, время работы алгоритма значительно возросло за счёт длительного времени предобработки данных (более 5 минут). Однако если сохранять предобработанные данне на диск между экспериментами, то можно нивелировать это время, и, более того, за счёт уменьшения размерности признакого пространства с 89000 до 40000.

Ещё одним методом улучшения качества может служить использование более сложных признаков, чем BagOfWords, например, Tfidf.

-	One-vs-All	All-vs-All	MN
Accuracy (Train/Test) $mindf = 2$	0.498/0.498	0.532/0.532	0.775/0.705
Accuracy (Train/Test) $mindf = 4$	0.487/0.485	0.522/0.519	0.775/0.706
Accuracy (Train/Test) $mindf = 4$	0.487/0.486	0.521/0.520	0.773/0.708

Таблица 3: Точность для различных алгоритмов мультиклассовой классификации после лемматизации, удаления стоп-слов с использованием Tfidf

Однако, как видно из таблицы 3, что данное изменение приводит к значительному падению качества в случае методов одни-против-всех и каждый-против-каждого. При этом для MN наблюдается значительный прирост качества.

Итоги

Можно отметить, что MN работает значительно лучше остальных алгоритмов даже с учётом небольшого числа итераций SGD. Как следствие, для итогового тестирования используются следующая модель: MN, обучаемая с помощью SGD ($\alpha = 0.1, \beta = 0.1$, размер батча 1). Все данные предварительно лемматизированы и из них удалены стоп слова. Затем, выделены признаки Tfidf. В итоге был получен результат 0.891/0.772 - точность на обучающей и тестовой выборке соответственно. При этом параметр $mindf$ почти не влияет на итоговый результат, что позволяет снизить размер признакового пространства до 10000 без потери качества, но с увеличением скорости на порядок.

Изучая объекты, на которых алгоритм допускает ошибки можно выделить следующие особенности:

1. Маленькая длина отзыва
2. Большое число спецсимволов и цифр

Выводы

Эксперименты, проведённые в этом отчёте показывают, что:

1. SGD работает стабильнее, менее чувствителен к выбору гиперпараметров, имеет более высокую скорость сходимости.
2. В целом, методы один-против-всех и каждый-против-каждого работают намного хуже MN. При этом, при снижении размерности признакового пространства, происходит значительное падение качества этих двух методов.