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

Градиентные методы обучения линейных моделей

Шаталов Н.А
кафедра ММП, группа 317
2017

Содержание

1 Введение.

В данной работе описаны этапы проведения экспериментов над градиентными методами обучения линейных моделей. В качестве линейных моделей будем рассматривать логистическую регрессию для задачи бинарной классификации, стратегии one-vs-all и all-vs-all для сведения задачи бинарной классификации к многоклассовой и мультиномиальную логистическую регрессию для задачи многоклассовой классификации.

2 Теоретическая часть

Пусть имеется обучающая выборка $X = \{(x_i, y_i)\}_{i=1}^{l}$, где $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{Y} = \{-1, 1\}$.

Алгоритм логистической регрессии для задачи бинарной классификации обучается минимизацией функции потерь, которая задается так:

$$Q(X, w) = \frac{1}{l} \sum_{i=1}^{l} \log\left(1 + e^{-y \odot Xw}\right) + \frac{\lambda_2}{2} \|w\|_2^2, \quad \text{где } \lambda_2 \text{ - коэффициент } L_2 \text{ регуляризатора.}$$

Градиент этой функции потерь выглядит так:

$$\nabla_w Q(X, w) = -\frac{1}{l} X^T \left(y \odot \frac{1}{1 + \exp(y \odot Xw)} \right) + \lambda_2 w.$$

Пусть теперь $\mathbb{Y} = \{1, \ldots, K\}$.

Алгоритм мультиномиальной регрессии для задачи многоклассовой классификации обучается минимизацией функции потерь:

$$Q(X, W) = -\frac{1}{l} \sum_{i=1}^{l} \log\left(\frac{\exp(\langle w_{y_i}, x_i \rangle)}{\sum_{k=1}^{K} \exp(\langle w_k, x_i \rangle)} \right) + \frac{\lambda_2}{2} \sum_{k=1}^{K} \|w_k\|_2^2.$$

Градиент этой функции потерь выглядит так:

$$\nabla_{w_j} Q(X, W) = \frac{1}{l} \sum_{i=1}^{l} \left(\frac{\exp(\langle w_j, x_i \rangle)}{\sum_{k=1}^{K} \exp(\langle w_k, x_i \rangle)} - [y_i = j] \right) x_i + \lambda_2 w_j.$$

Покажем, что при $K = 2$ задача мультиномиальной логистической регрессии сводится к бинарной логистической регрессии:

$$Q(X, W) = -\frac{1}{l} \sum_{i=1}^{l} \log\left(\frac{\exp(\langle w_{y_i}, x_i \rangle)}{\sum_{k=1}^{2} \exp(\langle w_k, x_i \rangle)} \right).$$

Сократим экспоненту в числителе, учитывая что $y_i \in \{-1, 1\}$:

$$\begin{aligned} Q(X, W) &= -\frac{1}{l} \sum_{i=1}^{l} \log\left(\frac{1}{1 + \exp(y_i \langle w_1 - w_2, x_i \rangle)} \right) = \\ &= -\frac{1}{l} \sum_{i=1}^{l} \log\left(\frac{1}{1 + \exp(-y_i \langle w, x_i \rangle)} \right) = \frac{1}{l} \sum_{i=1}^{l} \log\left(1 + \exp(-y_i \langle w, x_i \rangle)\right). \end{aligned}$$

3 Эксперименты

3.1 Сравнение полного и стохастического градиентного спуска

Эксперименты проводились на датасете REAL-SIM, который состоит из 72309 документов. Размер батча в стохастическом градиентном спуске равен 1, поэтому для этого метода были увеличены максимальное количество итераций (max_iter) и параметр допустимого отклонения *tolerance*, так как разница функции потерь между двумя итерациями может принимать намного меньшее значение, чем при полном градиентном спуске. Значения всех параметров приведены в таблице 1. [λ_2 - коэффициент L_2 регуляризатора, α и β - параметры темпа обучения μ_k (шаг метода на k-ой итерации), который равен $\mu_k = \alpha/k^\beta$].

	Полный	Стохастический
max_iter	10000	140000
λ_2	1e-5	1e-5
α	1	1
β	0	0
tolerance	1e-5	1e-8

Таблица 1: Параметры алгоритмов градиентного спуска

На рис. 2 показаны зависимости функции потерь и точности от номера итерации метода. Заметим, что при размере обучающей выборки 70000 объектов и размере батча 1 одна эпоха в стохастическом методе проходит за 70000 итераций. Как видно из графиков, стохастический градиентный спуск не уступает по качеству полному, так как сходится почти в той же точке. Однако, функция потерь и точность в стохастическом градиентном спуске менее устойчивы, это видно по разбросам на графиках.

На рис. 3 показаны зависимости функции потерь и точности от времени работы методов. Как видно, стохастический градиентный спуск работает быстрее полного.

Точность в обоих методах резко возрастает из-за удачного выбора начальных значений весов (в данном случае веса инициализируются нулевыми значениями).

Если начальные веса взять из нормального распределения с математическим ожиданием $\mu = 0$ и дисперсией $\sigma^2 = 10$, то, как видно на рис. 1 точность будет возрастать гораздо медленнее. Полному стохастическому градиентному спуску не хватило 10000 итераций, чтобы дойти до минимума функции потерь, поэтому его точность намного меньше. Скорость сходимости можно ускорить, исправив темп обучения.

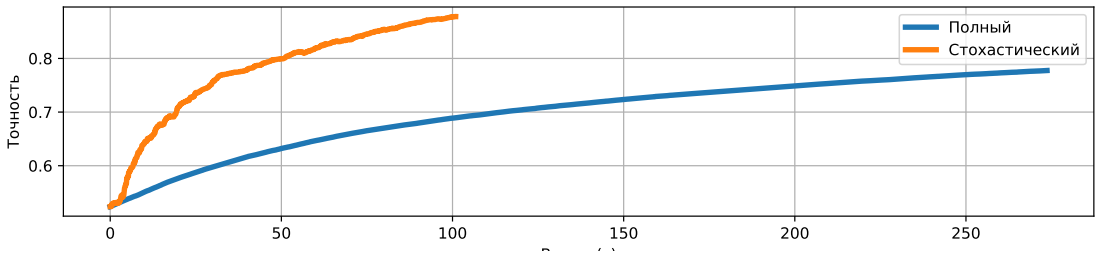


Рис. 1: Зависимость точности от времени при ненулевых начальных весах

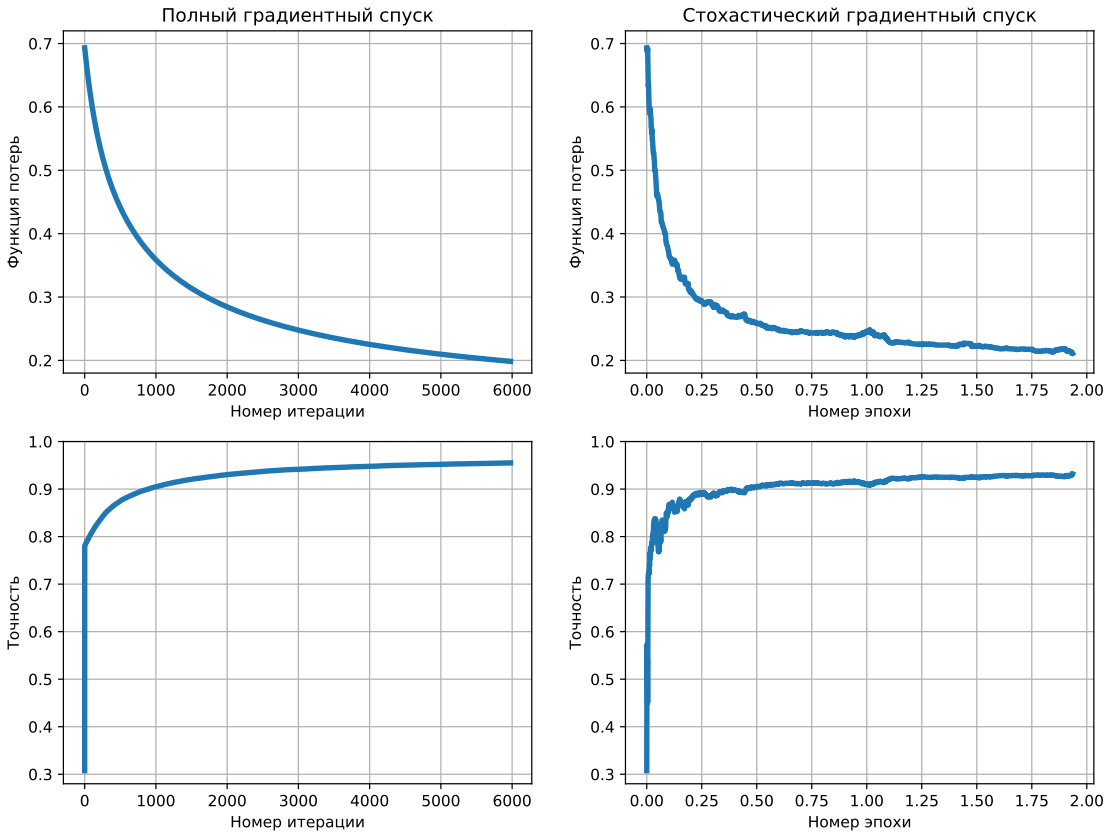


Рис. 2: Зависимость функции потерь и точности от номера итерации

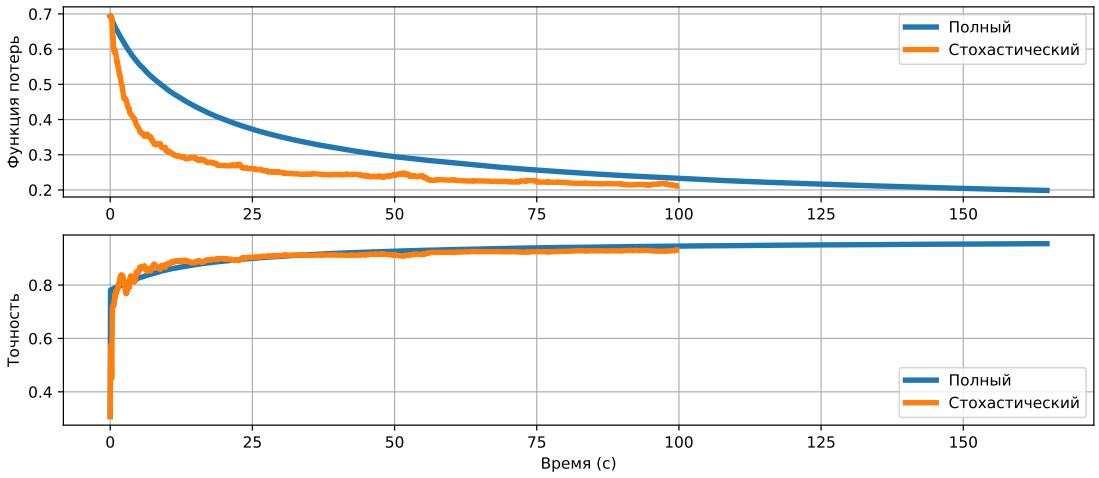


Рис. 3: Зависимость функции потерь и точности от времени

3.2 Влияние темпа обучения на качество алгоритма

Посмотрим, как влияют параметры α_k и β_k в шаге алгоритмов полного и стохастического градиентных спусков. Остальные параметры алгоритмов такие же, как и в прошлом эксперименте. Напомним, что темп обучения равен:

$$\mu_k = \frac{\alpha}{k^\beta}.$$

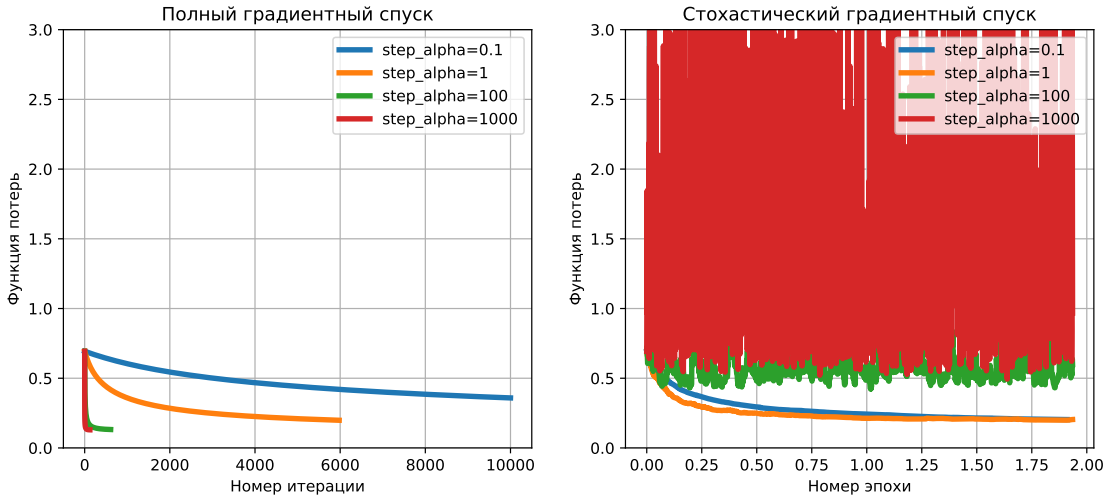


Рис. 4: Зависимость функции потерь от номера итерации при различном параметре α

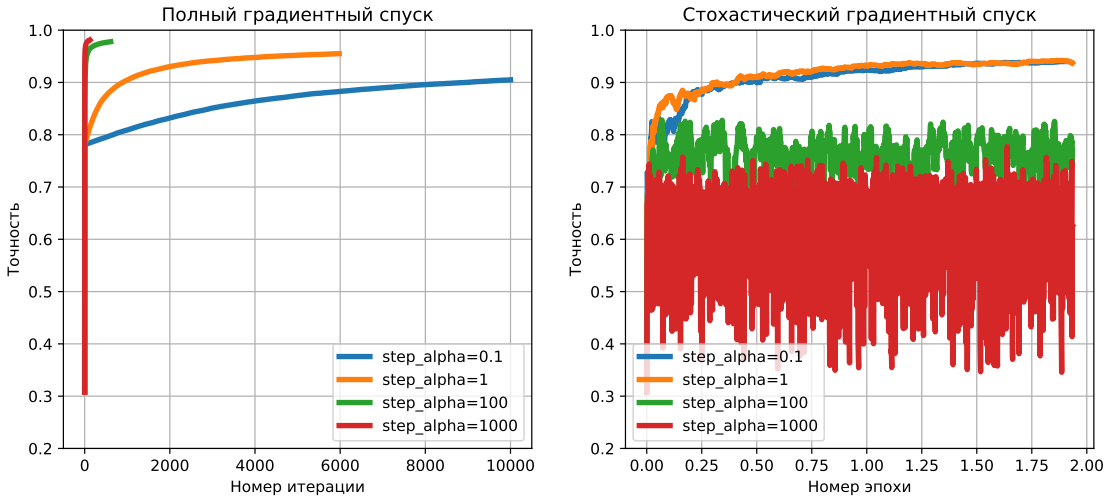


Рис. 5: Зависимость точности от номера итерации при различном параметре α

Заметим, что чем больше α, тем больше шаг в каждой итерации алгоритма. Из рис. 4 и рис. 5 видно, что увеличение параметра α положительно влияет на полный градиентный спуск и ускоряет процесс обучения. Тем временем, большие параметры α делают обучение

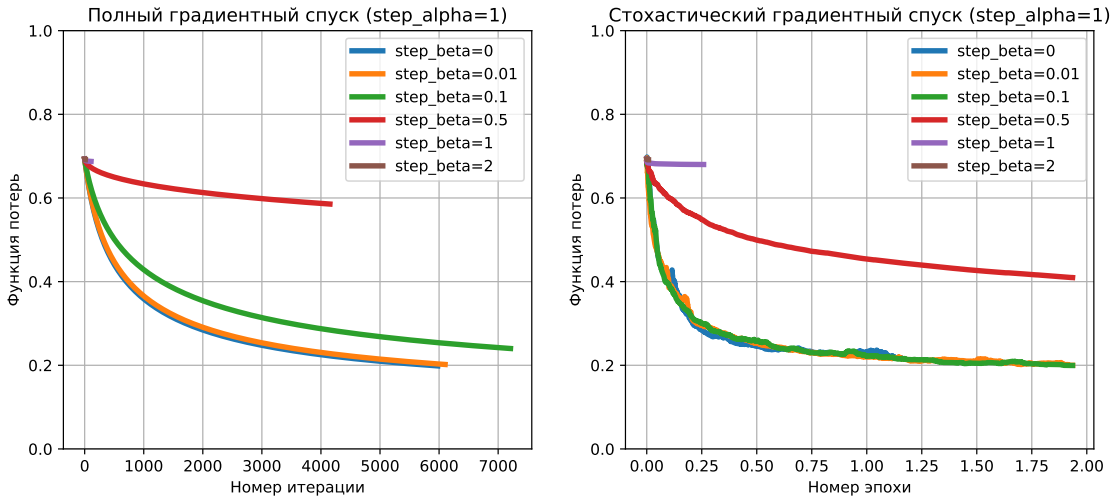


Рис. 6: Зависимость функции потерь от номера итерации при различном параметре β

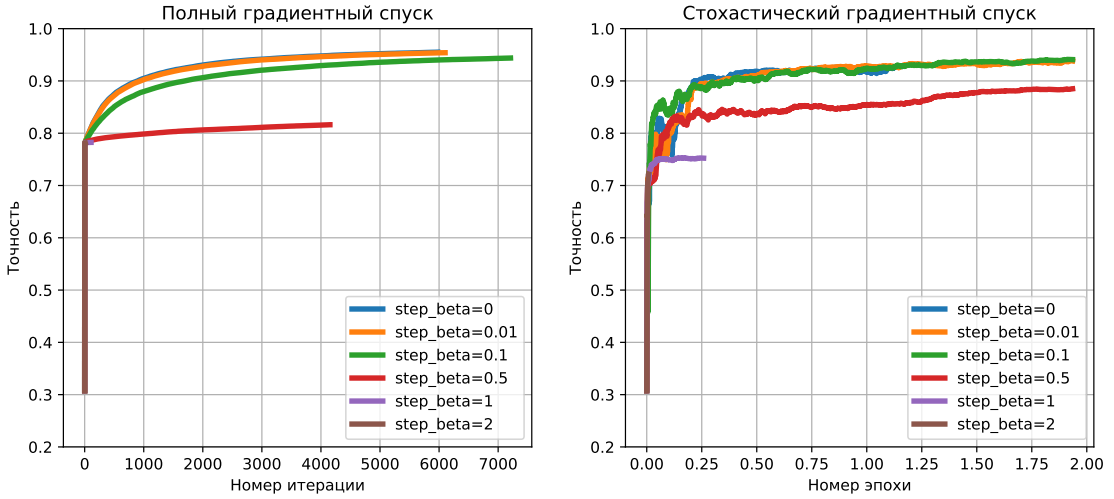


Рис. 7: Зависимость точности от номера итерации при различном параметре β

с помощью стохастического градиентного спуска неустойчивым, так как градиент считается только по одному элементу.

Заметим, что чем больше параметр β, тем короче становится шаг с каждой следующей итерацией. На рис. 6 и рис. 7 можно видеть, что при большом β алгоритм останавливает обучение слишком рано, из-за маленькой разницы между двумя итерациями.

3.3 Влияние случайности выбора объектов в методе стохастического градиентного спуска

Посмотрим, как будет происходить процесс обучения при различных random_seed алгоритма стохастического градиентного спуска.

Как можно видеть на рис. 8, алгоритмы имеют заметную разницу в качестве в начале обу-

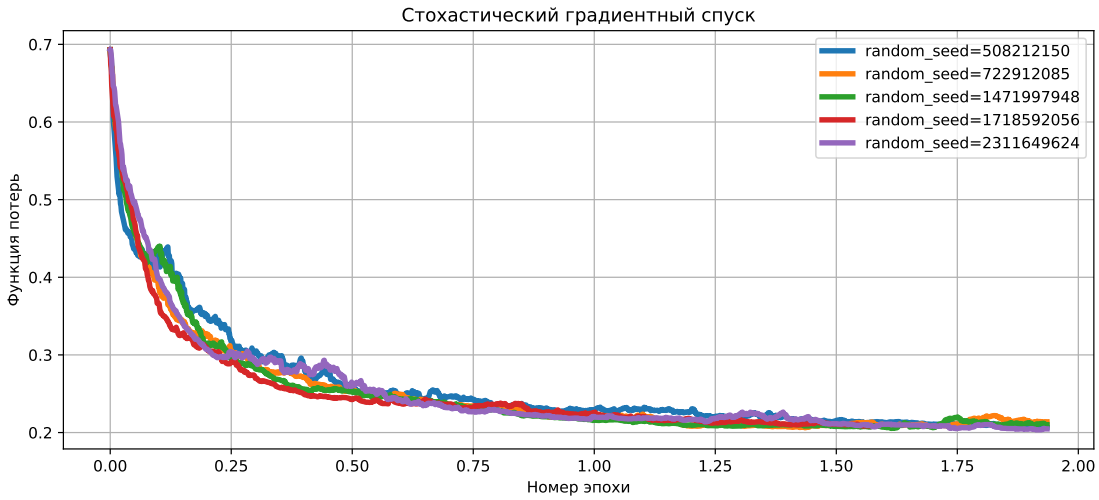


Рис. 8: Зависимость функции потерь от эпохи при разной случайности выбора объектов

чения, но стабилизируются в конце, и при разной случайности алгоритмы сходятся примерно в одной и той же точке.

3.4 Влияние размера подвыборки в методе стохастического градиентного спуска

Посмотрим, как влияет размер подвыборки в методе стохастического градиентного спуска на процесс обучения.

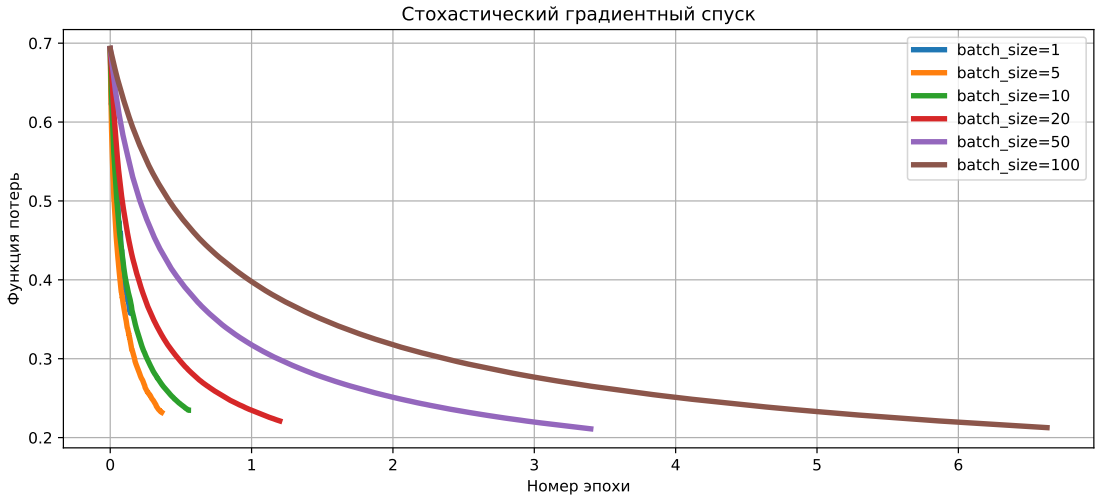


Рис. 9: Зависимость функции потерь от номера эпохи при разных размерах подвыборок

На рис. 9 видно, что при размере подвыборки 1 алгоритм был неустойчив и недообучился. При остальных параметрах метод сходился устойчиво, и дошел примерно до одной и той же точки.

Размер подвыборки	Кол-во эпох	Точность	Функция потерь
1	0.1393	0.8551	0.3559
5	**0.0332**	0.8318	0.4944
10	0.4658	0.9400	0.2467
20	0.8419	0.9380	0.2494
50	4.0852	**0.9546**	**0.1994**
100	6.5925	0.9512	0.2128

Таблица 2: Качество алгоритмов стохастического градиентного спуска при различных размерах подвыборки

Из таблицы 2 можно заметить, что качество улучшается при увеличении размера подвыборки (размер 50 ненамного лучше размера 100, из-за случайности выбора подвыборки). Однако с повышением размера подвыборки увеличивается время обучения. Заметим, что при размере подвыборки 1 присутствует сильное отличие в качестве, чем при размере подвыборки 5. Это из-за того, что обучение по одному элементу обучающей выборки неустойчиво, поэтому оно происходит менее качественно, и даже дольше.

3.5 Сравнение разных стратегий мультиклассовой классификации

Сравним, как работают стратегии «один против всех», «все против всех» и алгоритм мультиномиальной регрессии для задачи многоклассовой классификации. Для сравнения были сгенерированы две выборки. Первая сделана так, что центры кластеров находятся близко друг к друг, при этом разброс в каждом из кластеров небольшой. Вторя создана так, что центры кластеров находятся далеко друг от друга, но имеют большой разброс. Обе выборки линейно неразделимы.

Результат можно увидеть на рис. 10.

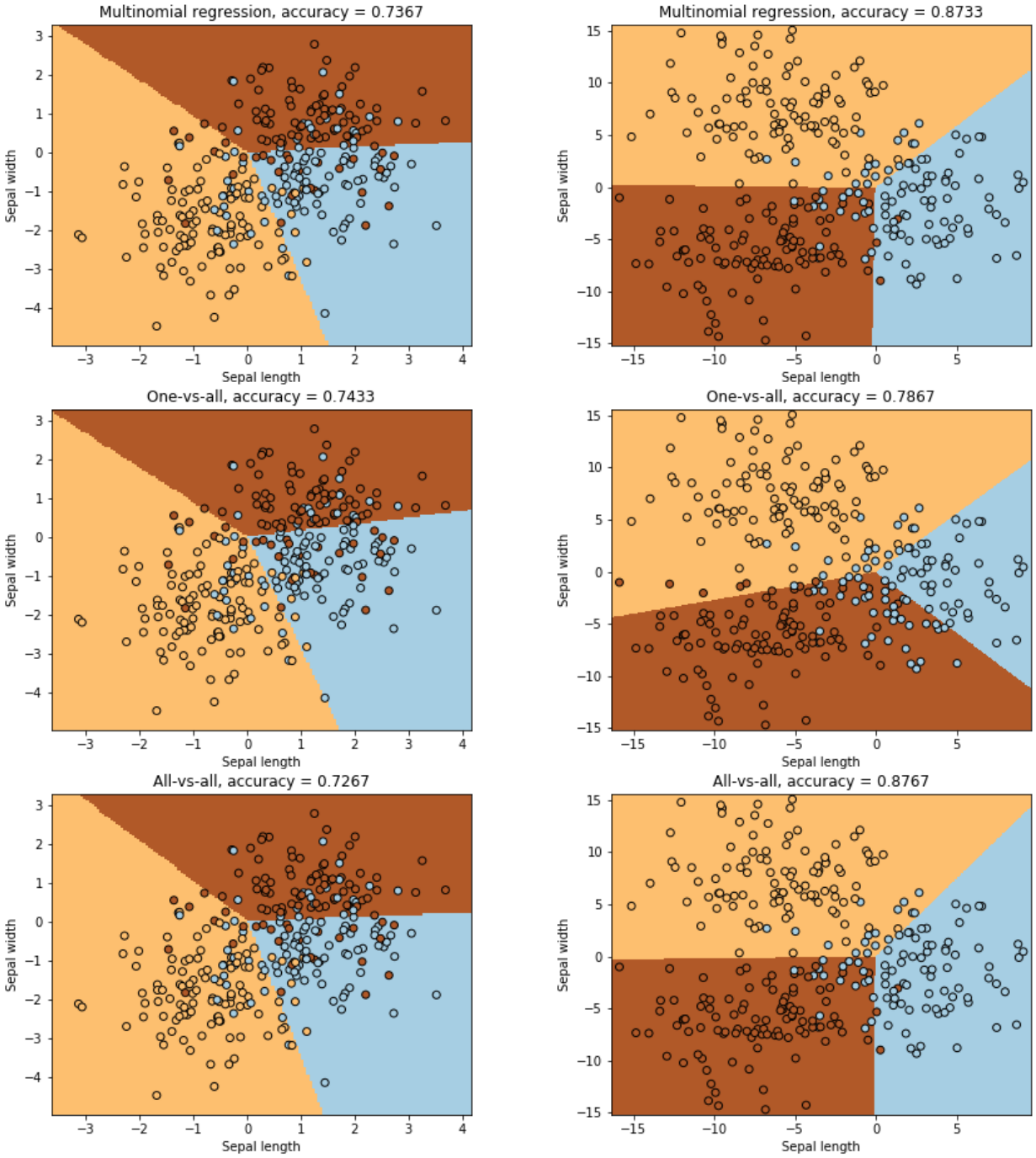
Как видно, стратегия «один против всех» лучше работает для пересекающихся, близко расположенных объектов (рис. 10a), а стратегия «все против всех» лучше работает для более лучше разделимых данных (рис. 10b).

Мультиномиальная логистическая регрессия показывает средний результат в обоих случаях, но имеет преимущество в обучении, так как оно занимает намного меньше времени (обучение одного классификатора вместо K или C_K^2).

3.6 Использование мультиномиальной логистической регрессии на текстовой коллекции

Дальнейшие эксперименты проводились на коллекции 20newsgroups, состоящем из 11314 документов. Для настройки параметров алгоритма обучающая выборка была разделена на отложенную и валидационную. Размер валидационной выборки равен 4000 документов, что составляет 30% обучающей выборки.

Как предобработка данных, все символы, не являющиеся буквами или цифрами, были заменены на пробелы, все буквы переведены в нижний регистр. Каждый документ преобразовывался сначала в вектор счетчика слов из коллекции, а после производилось tf-idf преобразование. Слова, вошедшие в валидационную, но не в отложенную выборку – игнорировались.



(a) Выборка №1

(b) Выборка №2

Рис. 10: Сравнение разных стратегий мультиклассовой классификации

Параметр	Значение
α	100
β	0.01
Размер подвыборки	100
max_iter	2000
λ_2	0

Таблица 3: Подобранные на валидационной выборке оптимальные параметры

В таблице 3 представлены оптимальные параметры, подобранные на валидационной выборке. Сделаем финальное предсказание на тестовой выборке, используя оптимальные параметры:

$$accuracy = 0.68.$$