Кузьмин Никита, ММП, 317.

Отчет по практическому заданию №2 "**Применение линейных моделей для определения токсичности комментария**".
Логистическая регрессия и градиентный спуск.

Содержание

1 Введение

В данном документе представлен отчет о проделанных экспериментах по практическому заданию №2, анализ результатов. Краткое описание задания: необходимо реализовать линейный классификатор с произвольной функцией потерь.

2 Теория

Пусть $X \in \mathbb{R}^{N \times d}$ - обучающая выборка, $y \in \mathbb{R}^N$ - ответы на обучающей выборке, причем $x_i \in \mathbb{R}^d$ - i-ая строка матрицы X и $y_i \in \{1, \dots, K\}$; $\lambda \in \mathbb{R}$ - коэффициент регуляризации, $w_k \in \mathbb{R}^d$ - набор весов, причем $k \in \{1, \dots, K\}$

2.1 Вывод формулы градиента функции потерь для задачи бинарной логистической регрессии

Пусть в данном случае: $y_i \in \{-1, 1\}, K = 1, w_1 = w$.
Функция потерь для задачи бинарной логистической регрессии имеет вид:

$$\mathcal{L}(X, y, w) = \frac{1}{N} \sum_{i=1}^{N} \ln(1 + e^{-w^T x_i y_i}) + \frac{\lambda}{2} w^T w$$

Найдем дифференциал $\mathcal{L}$ по w:

$$d_w \mathcal{L}(X, y, w) = \frac{1}{N} \sum_{i=1}^{N} \frac{-y_i d_w (e^{-w^T x_i y_i})}{1 + e^{-w^T x_i y_i}} + \lambda d_w w = \frac{1}{N} \sum_{i=1}^{N} \frac{-y d_w^T w x (e^{-w^T x_i y_i})}{1 + e^{-w^T x_i y_i}} + \lambda d_w w^T w$$

Тогда:

$$\nabla_w \mathcal{L}(X, y, w) = \frac{1}{N} \sum_{i=1}^{N} \frac{-y_i w x_i (e^{-w^T x_i y_i})}{1 + e^{-w^T x_i y_i}} + \lambda w$$

$$\nabla_w \mathcal{L}(X, y, w) = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{-y_i x_i (1 + e^{-w^T x_i y_i})}{1 + e^{-w^T x_i y_i}} + y x \sigma(w^T x_i y_i) \right) + \lambda w$$

$$\nabla_w \mathcal{L}(X, y, w) = \frac{1}{N} \sum_{i=1}^{N} y_i x_i (-1 + \sigma(w^T x_i y_i) + \lambda w = \frac{1}{N} \sum_{i=1}^{N} -y_i x_i \sigma(-w^T x_i y_i) + \lambda w$$

2.2 Вывод формулы градиента функции потерь для задачи многоклассовой (мультиноминальной) логистической регрессии

Функция потерь для задачи многоклассовой логистической регрессии выглядит следующим образом:

$$\mathcal{L}(X, y, w) = -\frac{1}{N} \sum_{i=1}^{N} \log \left(\frac{e^{<w_{y_i}, x_i>}}{\sum_{k=1}^{K} e^{<w_{y_k}, x_i>}} \right) + \frac{\lambda}{2} \sum_{k=1}^{K} ||w_k||_2^2$$

Тогда градиент функции потерь можно представить в виде:

$$\nabla_\omega \mathcal{L}(X, \omega) = \left(\frac{\partial \mathcal{L}}{\partial \omega_1}, \dots, \frac{\partial \mathcal{L}}{\partial \omega_K} \right)$$

Далее рассмотрим два случая:

- $y_i \neq w_j$

$$\frac{d\mathcal{L}}{dw_j} = -\frac{\sum_{k=1}^{K} e^{(\langle w_k, x_i\rangle)}}{e^{(\langle w_{y_i}, x_i\rangle)}} - \frac{e^{(\langle w_{y_i}, x_i\rangle)} e^{\left(\langle w_j, x_i\rangle\right)} x_i}{\left(\sum_{k=1}^{K} e^{(w_k, x_i)}\right)^2} + \lambda w_j =$$
$$x_i \frac{e^{\left(\langle w_j, x_i\rangle\right)}}{\sum_{k=1}^{K} e^{(\langle w_k, x_i\rangle)}} + \lambda w_j$$

- $y_i = w_j$

$$\frac{d\mathcal{L}}{dw_j} = -\frac{\sum_{k=1}^{K} e^{(\langle w_k, x_i\rangle)} e^{\left(\langle w_j, x_i\rangle\right)} x_i \sum_{k=1}^{K} e^{(\langle w_k, x_i\rangle)} - e^{\left(\langle w_j, x_i\rangle\right)} e^{\left(\langle w_j, x_i\rangle\right)} x_i}{e^{\left(\langle w_j, x_i\rangle\right)}} \frac{\sum_{k=1}^{K} e^{(\langle w_k, x_i\rangle)} e^{\left(\langle w_j, x_i\rangle\right)} x_i}{\left(\sum_{k=1}^{K} e^{(\langle w_k, x_i\rangle)}\right)^2} +$$
$$+\lambda w_j = -x_i + \frac{e^{\left(\langle w_j, x_i\rangle\right)}}{\sum_{k=1}^{K} e^{(\langle w_k, x_i\rangle)}} x_i + \lambda w_j = x_i \left(-1 + \frac{e^{\left(\langle w_j, x_i\rangle\right)}}{\sum_{k=1}^{K} e^{(\langle w_k, x_i\rangle)}}\right) + \lambda w_j$$

Таким образом получим компактное выражение:

$$\frac{d\mathcal{L}}{dw_j} = x_i \left(-[y_i = w_j] + \frac{e^{(\langle w_j, x_i\rangle)}}{\sum_{k=1}^{K} e^{(\langle w_k, x_i\rangle)}}\right) + \lambda w_j$$

2.3 Анализ бинарной и многоклассовой логистической регрессии

Покажем, что при $K = 2, y_i \in \{-1, 1\}, \forall i$ формула функции потерь для многоклассовой логистической регрессии сводится к формуле для бинарной логистической регрессии:

$$\mathcal{L}(X, y, w) = -\frac{1}{N} \sum_{i=1}^{N} \log\left(\frac{e^{<w_{y_i}, x_i>}}{e^{<w_1, x_i>} + e^{<w_2, x_i>}}\right) = -\frac{1}{N} \sum_{i=1}^{N} \log\left(\frac{1}{1 + e^{-y_i <w_2 + w_1, x_i>}}\right)$$

Тогда, если положить $w = w_2 + w_1$, получается формула функции потерь для бинарной логистической регрессии:

$$\mathcal{L}(X, y, w) = \frac{1}{N} \sum_{i=1}^{N} \log\left(1 + e^{-w^T x_i y_i}\right)$$

3 Эксперименты

В этом блоке приведены все обязательные эксперименты, которые изложены в формулировке задания. Все эксперименты проводились на упрощенном датасете (рассматривается задача бинарной классификации) из соревнования **Toxic Comment Classification Challenge**, в котором нужно определить токсичность комментария.

Стандартный дизайн эксперимента:

- Оценка качества и подбор параметров модели проводились на каждой эпохе с помощью отложенной тренировочной выборки (30%). Все графики ниже построены по значениям accuracy, посчитанным на отложенной выборке.
- В тренировочную выборку был добавлен признак, состоящий из всех единиц, который позволяет учитывать смещение (**bias**). Было решено не использовать смещение в $L2$-регуляризации, чтобы даже при плохом выборе коэффициента регуляризации решающая гиперплоскость не вырождалась в 0.
- В стохастическом градиентном спуске проверяется критерий останова на каждой эпохе (не итерации).
- Стандартные параметры экспериментов (если не оговорено обратное):
  - step_alpha = 1
  - step_beta = 0.001

- batch_size = 4096
- tolerance = 1e-5
- l2_coef = 1e-5
- max_iter = 1500

3.1 Исследование поведения градиентного спуска

Обновления весов модели при использовании градиетного спуска происходит по следующей формуле:

$$w_t = w_{t-1} - \frac{\alpha}{t^\beta} \times \frac{1}{N} \times \frac{1}{N} \sum_{i=1}^{N} \nabla_w \mathcal{L}(x_i, y_i | w_{t-1}), \tag{1}$$

где t - номер итерации, β - **step_beta**, $\nabla_w \mathcal{L}(x_i, y_i | w_{t-1})$ - градиент функции потерь.

3.1.1 Параметр размера шага step_alpha

Параметр **step_alpha** (α) используется в градиентном спуске при обновлении весов в формуле 1. Рассмотрим следующие зависимости при разных значениях параметра **step_alpha**:

1. зависимость значения функции потерь от реального времени работы метода
2. зависимость точности (accuracy) от реального времени работы метода
3. зависимость значения функции потерь от итерации метода
4. зависимость точности (accuracy) от итерации метода

Соответствующие графики приведены на: рис. 1, 2, 3, 4.

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

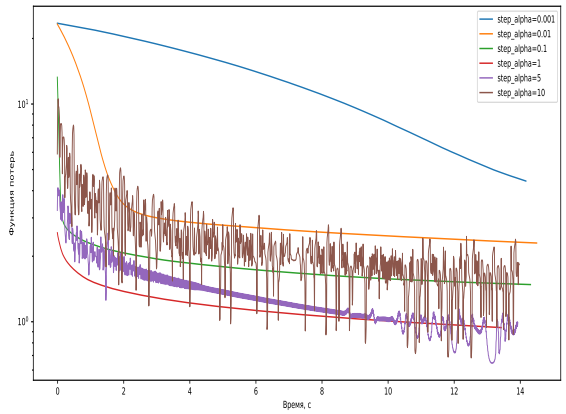


Рис. 2: Зависимость значения точности (accuracy) от реального времени работы градиентного спуска

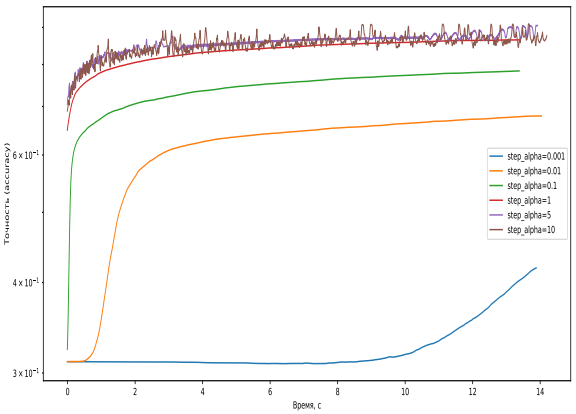


Рис. 3: Зависимость значения функции потерь от итерации метода градиентного спуска

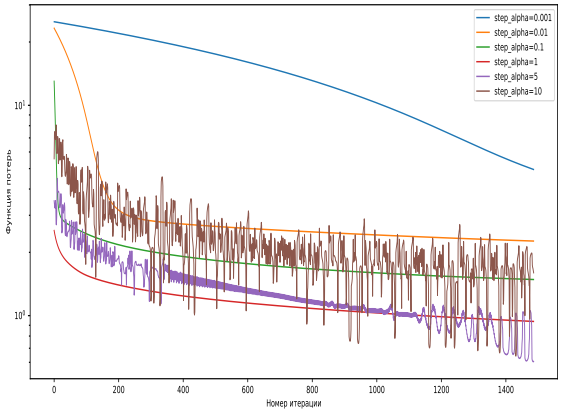
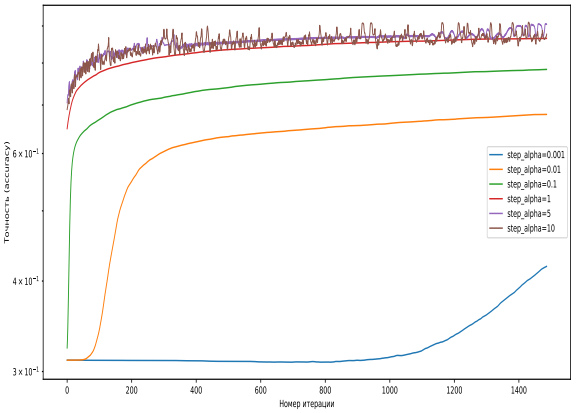


Рис. 4: Зависимость значения точности (accuracy) от итерации метода градиентного спуска



Из графиков видно, что при значениях α, близких к нулю алгоритму нужно больше времени для сходимости. С другой стороны, если значения слишком большие, то алгоритм становится крайне не стабильным.

3.1.2 Параметр размера шага step_beta

Параметр **step_beta** (β) используется в градиентном спуске при обновлении весов в формуле 1. Аналогично предыдущему пункту рассмотрим зависимости из 3.1.1 при разных значениях параметра **step_beta** и проанализруем соответсвующие графики, представленные на рис. 5, 6, 7, 8.

Рис. 5: Зависимость значения функции потерь от реального времени работы градиентного спуска

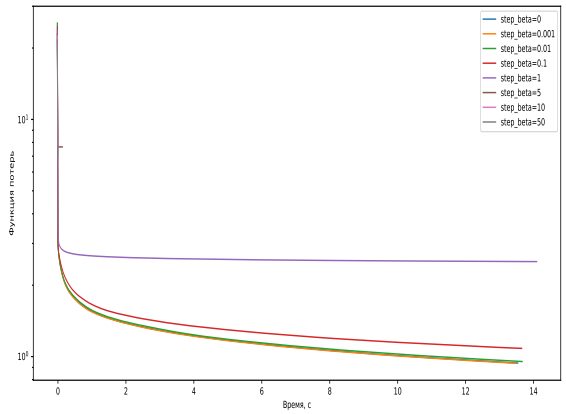


Рис. 6: Зависимость значения точности (accuracy) от реального времени работы градиентного спуска

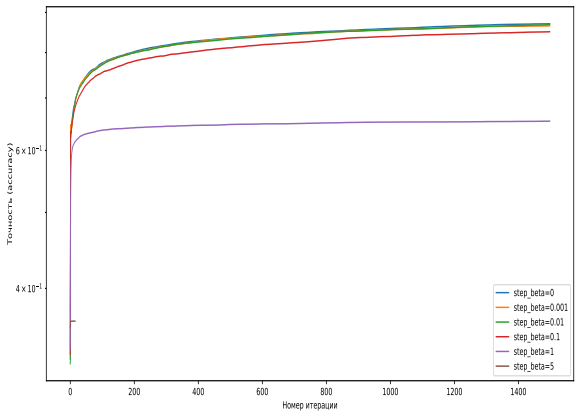


Рис. 7: Зависимость значения функции потерь от итерации метода градиентного спуска

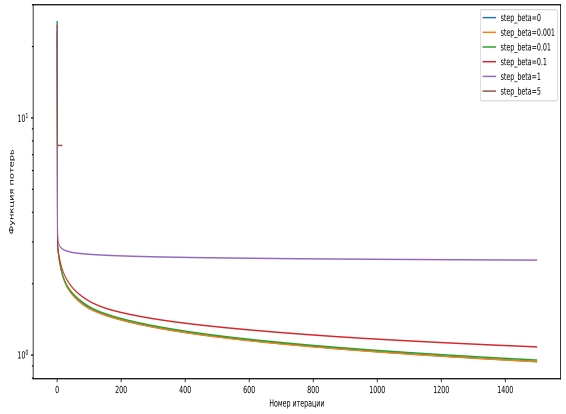
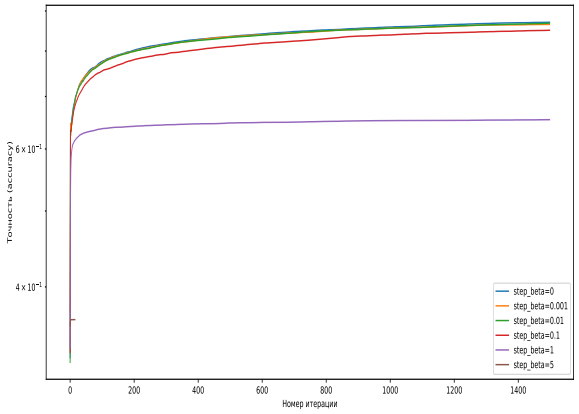


Рис. 8: Зависимость значения точности (accuracy) от итерации метода градиентного спуска



Из графиков можно заметить, что значения **step_beta**, близкие к 0, приводят к одинаковому качеству. С увеличением же параметра **step_beta** значение точности уменьшается. При **step_beta** $= 5$ изменение функции потерь так мало, что критерий остановки срабатывает до первых 200 итераций.

3.1.3 Начальное приближение w_0

Начальное приближение нужно для инициализации весов модели. В данной работе были рассмотрены следующие варианты задания w_0:

- нулевой вектор
- вектор с координатами из $U(0, 1)$
- вектор с координатами из $U(100, 500)$
- вектор с координатами из $U(1000, 5000)$
- вектор с координатами из $U(10000, 50000)$
- вектор с координатами из $N(0, 1)$
- вектор с координатами из $N(0.5, 0.5)$

Графики зависимостей 3.1.1 представлены на рис. 9, 10, 11, 12.

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

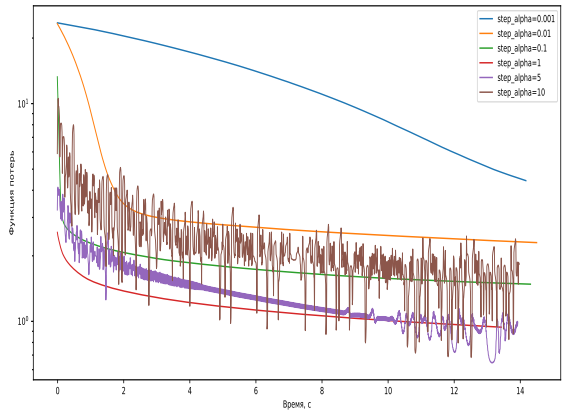


Рис. 10: Зависимость значения точности (accuracy) от реального времени работы стохастического градиентного спуска

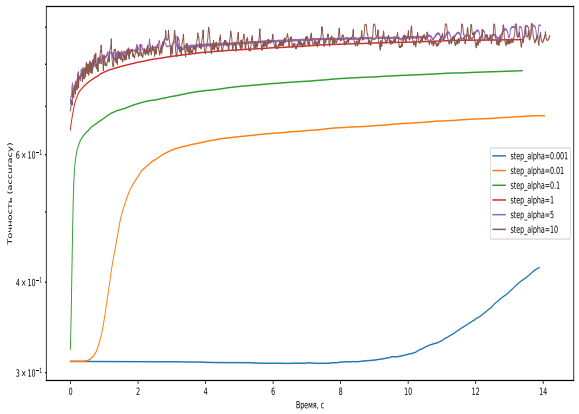


Рис. 11: Зависимость значения функции потерь от итерации метода стохастического градиентного спуска

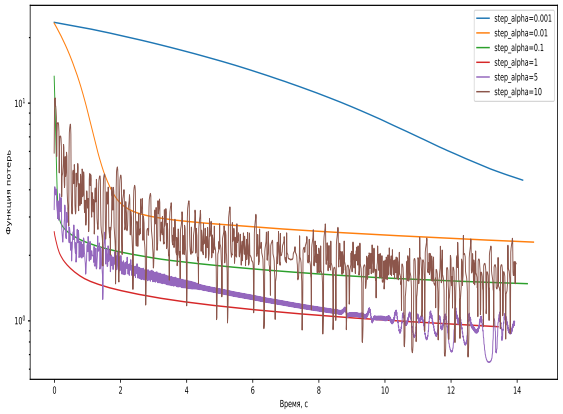
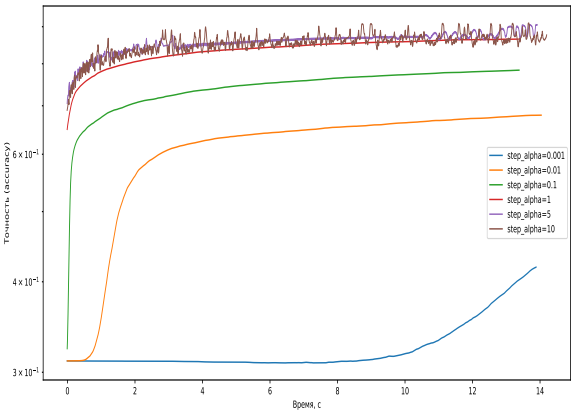


Рис. 12: Зависимость значения точности (accuracy) от итерации метода стохастического градиентного спуска



3.2 Исследование поведения стохастического градиентного спуска

Обновления весов модели при использовании стохастического градиетного спуска происходит по следующей формуле:

$$w_t = w_{t-1} - \frac{\alpha}{t^\beta} \times \frac{1}{|I|} \times \sum_{i \in I} \nabla_w \mathcal{L}(x_i, y_i | w_{t-1}), \tag{2}$$

где t - номер итерации, β - **step_beta**, I - некоторое подможнество индексов тренировочной выборки, $\nabla_w \mathcal{L}(x_i, y_i | w_{t-1})$ - градиент функции потерь.

3.2.1 Параметр размера шага step_alpha

Параметр **step_alpha** (α) используется в стохастическом градиентном спуске при обновлении весов в формуле 2. Рассмотрим следующие зависимости при разных значениях параметра **step_alpha**:

1. зависимость значения функции потерь от реального времени работы метода
2. зависимость точности (accuracy) от реального времени работы метода
3. зависимость значения функции потерь от эпохи метода
4. зависимость точности (accuracy) от эпохи метода

Соответствующие графики приведены на: рис. 13, 14, 15, 16.

Рис. 13: Зависимость значения функции потерь от реального времени работы градиентного спуска

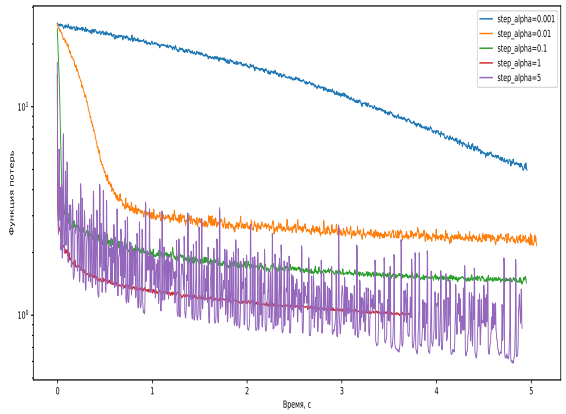


Рис. 14: Зависимость значения точности (accuracy) от реального времени работы градиентного спуска

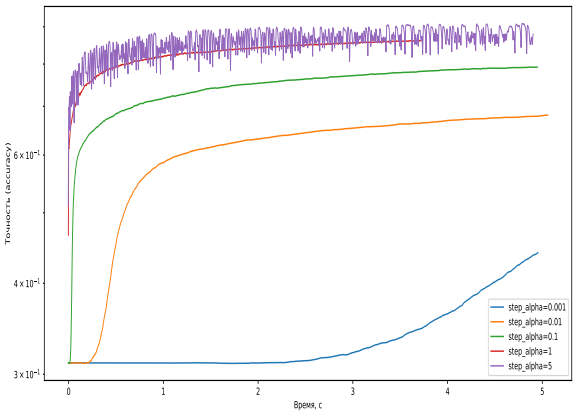


Рис. 15: Зависимость значения функции потерь от эпохи метода градиентного спуска

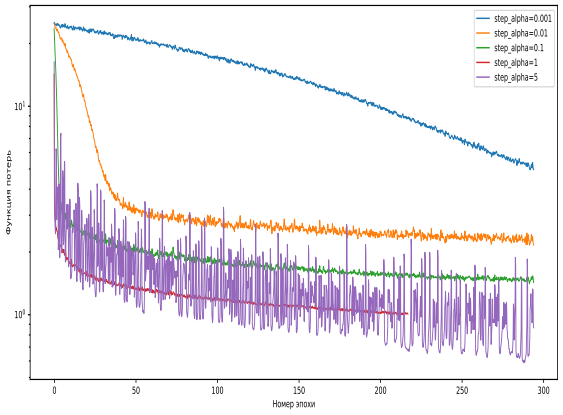
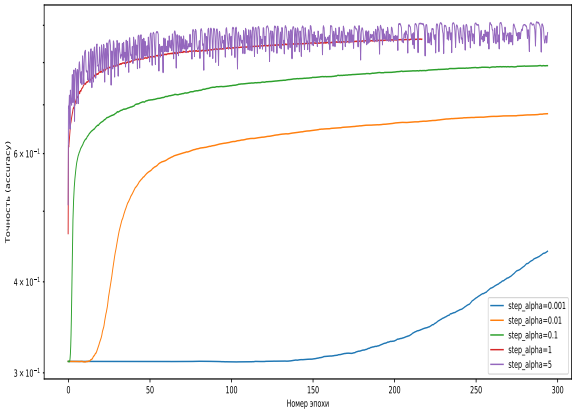


Рис. 16: Зависимость значения точности (accuracy) от эпохи метода градиентного спуска



На графиках просматривается аналогичная ситуация, что и с градиентным спуском: при значениях **step_alpha**, близких к 0 возникает эффект недообучения, а при больших - появляется нестабильность кривой обучения, но есть точки, в которых достигается наивысшая точность. В таком случае можно производить сохранение весов модели на итерации со значением наилучшей точности. Возможно, такой метод поможет справиться с нестабильностью.

3.2.2 Параметр размера шага step_beta

Параметр **step_beta** (β) используется в градиентном спуске при обновлении весов в формуле 1. Аналогично предыдущему пункту рассмотрим зависимости из 3.2.1 при разных значениях параметра **step_beta** и проанализируем соответсвующие графики, представленные на рис. 17, 18, 19, 20.

Рис. 17: Зависимость значения функции потерь от реального времени работы градиентного спуска

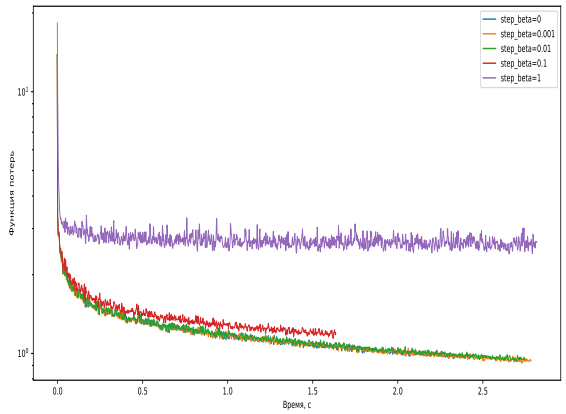


Рис. 18: Зависимость значения точности (accuracy) от реального времени работы градиентного спуска

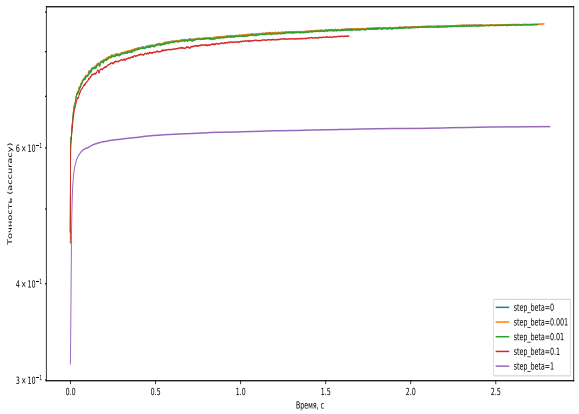


Рис. 19: Зависимость значения функции потерь от эпохи метода градиентного спуска

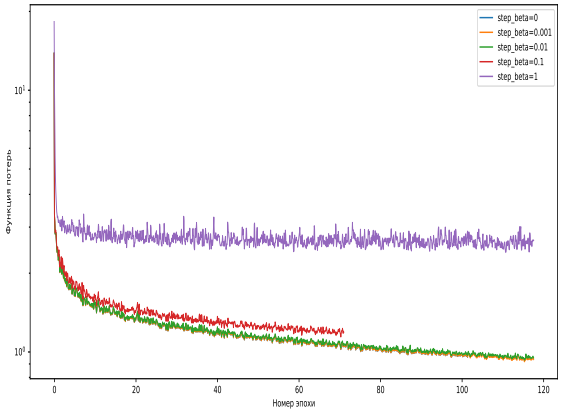
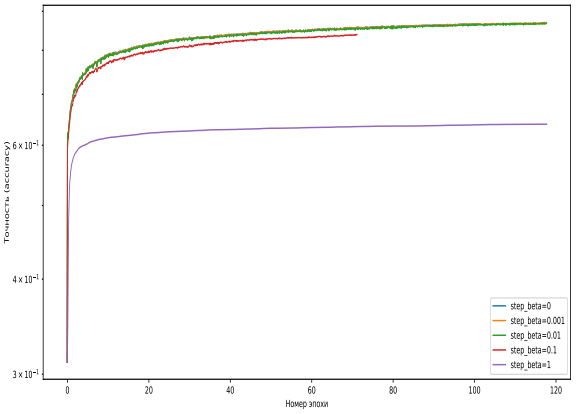


Рис. 20: Зависимость значения точности (accuracy) от эпохи метода градиентного спуска



Аналогично обычному градиентному спуску - с увелничением значения **step_beta** увеличивается значение функции потерь и уменьшается точность (accuracy).

3.2.3 Размер подвыборки batch_size

Размер подвыборки определяет количество элементов тренировочной выборки, которые будут использованы для подсчета градиента.

Соответствующие графики приведены на рис. 21, 22, 23, 24.

Рис. 21: Зависимость значения функции потерь от реального времени работы градиентного спуска

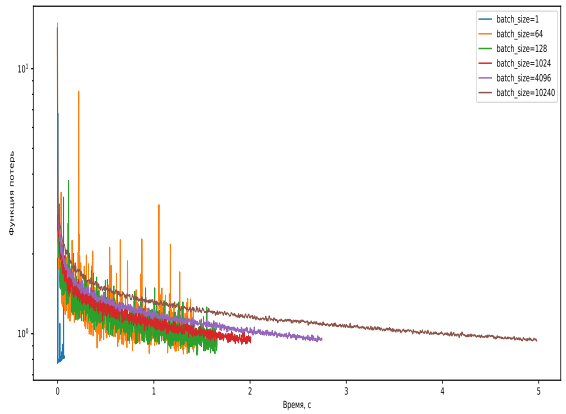


Рис. 22: Зависимость значения точности (accuracy) от реального времени работы градиентного спуска

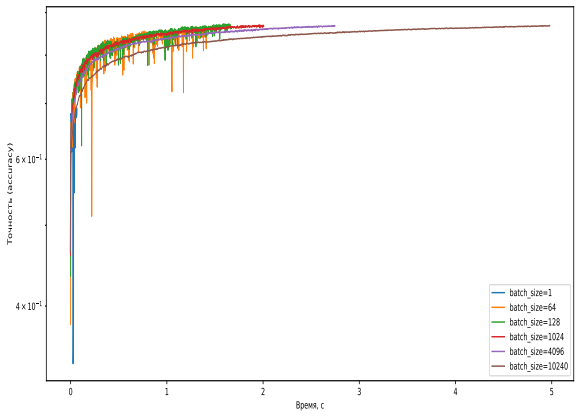


Рис. 23: Зависимость значения функции потерь от эпохи метода градиентного спуска

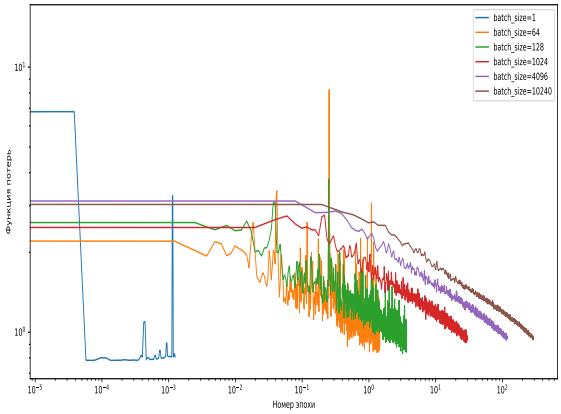
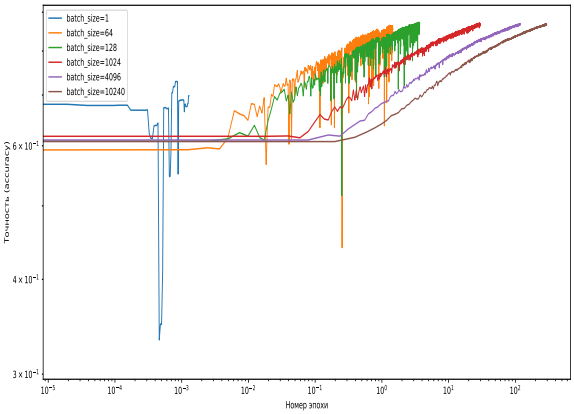


Рис. 24: Зависимость значения точности (accuracy) от эпохи метода градиентного спуска



Можно заметить сильное увеличение дисперсии значений функции потерь с уменьшением размера подвыборки. Это объясняется тем, что при изменении весов на каждой итерации используется подвыборка для вычисления градиента, то есть приближенное значение, которое может сильно меняться от итерации к итерации, если размер подвыборки достаточно маленький.

3.2.4 Начальное приближение w_0

Начальное приближение нужно для инициализации весов модели. В данной работе были рассмотрены следующие варианты задания w_0:

- нулевой вектор
- вектор с координатами из $U(0, 1)$
- вектор с координатами из $U(100, 500)$
- вектор с координатами из $U(1000, 5000)$
- вектор с координатами из $U(10000, 50000)$
- вектор с координатами из $N(0, 1)$

- вектор с координатами из $N(0.5, 0.5)$

Графики зависимостей 3.2.1 представлены на рис. 25, 26, 27, 28.

Рис. 25: Зависимость значения функции потерь от реального времени работы градиентного спуска

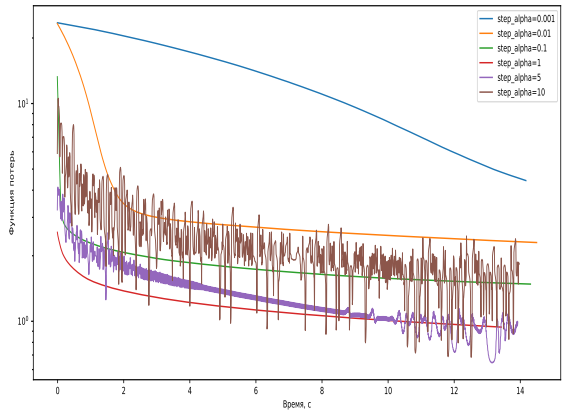


Рис. 26: Зависимость значения точности (accuracy) от реального времени работы градиентного спуска

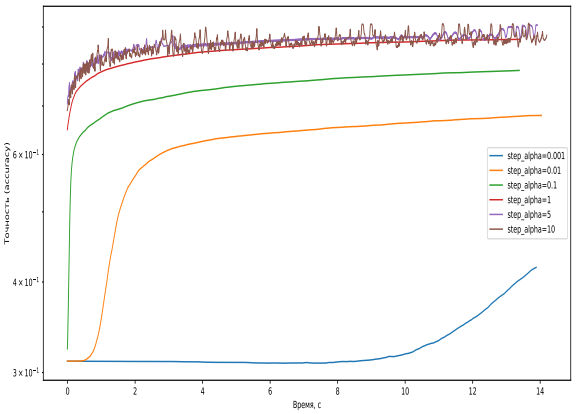


Рис. 27: Зависимость значения функции потерь от эпохи метода градиентного спуска

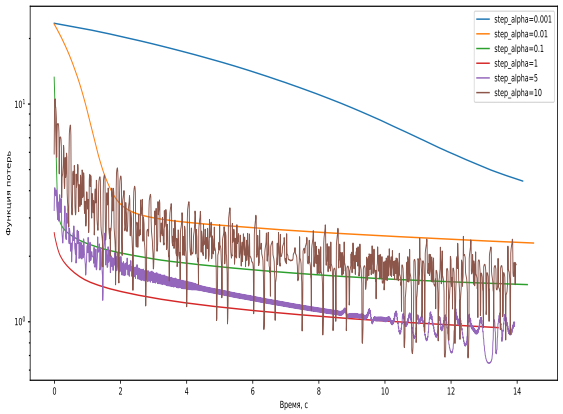
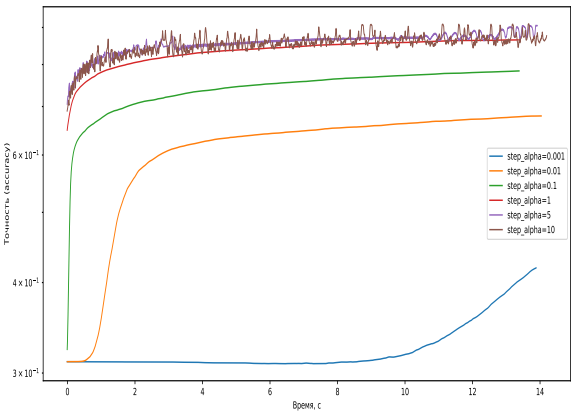


Рис. 28: Зависимость значения точности (accuracy) от эпохи метода градиентного спуска



Из графиков видно, что лучшим вариантом инициализации весов оказался нулевой вектор.

3.3 Сравнение градиентного спуска и стохастического градиентного спуска

В данном разделе проведено сравнение методов по трем характеристикам:

- значения функции потерь
- точность (accuracy)
- время работы метода

Результаты экспериментов приведены на рис. 29, 30 и в таблице 1

Рис. 29: Зависимость значения функции потерь от реального времени работы обычного и стохастического градиентного спуска

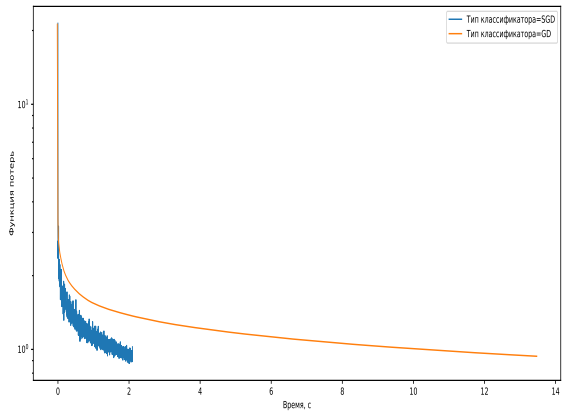


Рис. 30: Зависимость значения точности (accuracy) от реального времени работы обычного и стохастического градиентного спуска

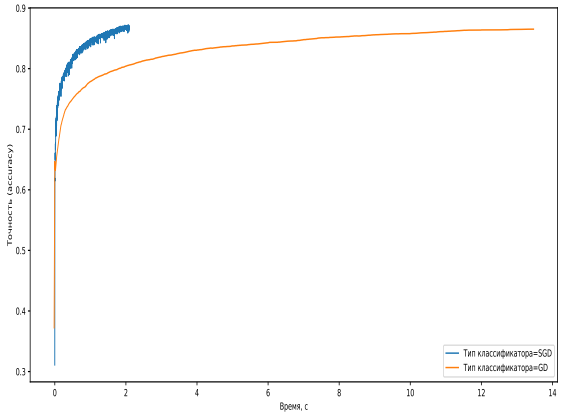


Таблица 1: Среднее время работы методов оптимизации

Метод	Среднее время работы, с
Градиентный спуск	8.852
Стохастический градиентный спуск	1.964

Из приведенных выше графиков видно, что стохастический градиентный спуск сходится быстрее, причем к более хорошему оптимуму, чем обычный градиентный спуск, но сходимость с бóльшей дисперсией значений функции потерь. Таблица 1 показывает, что стохастический градиентный спуск работает в несколько раз быстрее.

3.4 Лемматизация и удаление стоп-слов

В этом эксперименте проводится лемматизация, в которой считается, что все слова в тексте - глаголы. Удаляются все стоп-слова. В таблице 2 приведено сравнение метода градиентого спуска до преобразования текста и после. Все измерения проведены для градиентного спуска с параметрами: $step\_alpha = 1$, $step\_beta = 0.001$.

Таблица 2: Влияние предобработки на качество алгоритма

Метод предобработки	Точность	Среднее время работы, с	Количество признаков
(1), (2)	0.8800	13.59	91840
(1), (2), (3), (4)	0.8766	13.21	83530
(1), (2), (3)	**0.9016**	**13.09**	83400

(1) - приведение к нижнему регистру

(2) - удаление всех символов, отличных от букв и цифр

(3) - лемматизация

(4) - удаление стоп-слов

Как видно из таблицы - с уменьшением признакового пространства уменьшается и время работы

алгоритма. Можно заметить интересную особенность: удаление стоп-слов понижает точность классификации. Есть предположение, что удаление стоп-слов - не самая лучшая идея в данной задаче, так как в них есть такие, как «aren't», «didn't», которые указывают на отрицание. Пример: you aren't nice -> you nice (после удаления стоп-слов).

3.5 Сравнение представлений BagOfWords и TF-IDF с различными значениями параметров min_df и max_df

В данном разделе рассматривается зависимость качества, времени работы алгоритма и размер признакового пространства от:

- использовалось представление BagOfWords или TF-IDF
- параметров min_df и max_df конструкторов

Графики зависимостей представлены на рис. 31, 32, 33

Рис. 31: Зависимость точности от типа представления

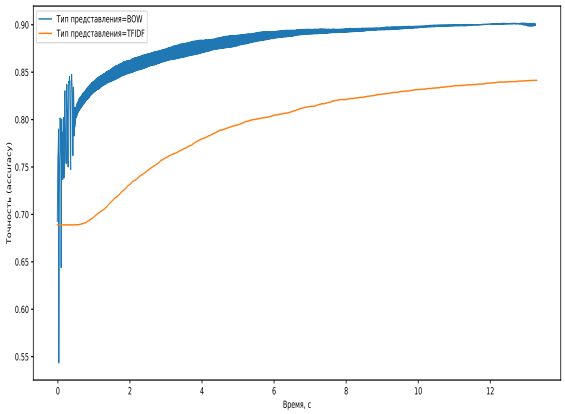


Рис. 32: Зависимость точности от параметра конструктора min_df

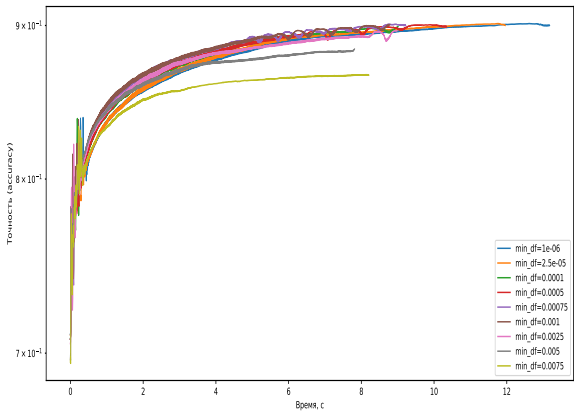


Рис. 33: Зависимость размера признакового пространства от параметра конструктора min_df

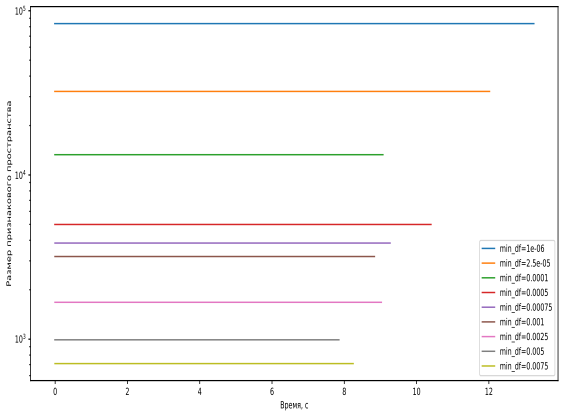


Рис. 34: Зависимость точности от типа представления

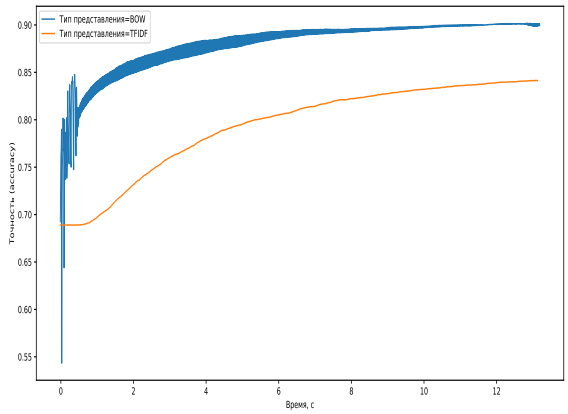


Рис. 35: Зависимость точности от параметра конструктора min_df

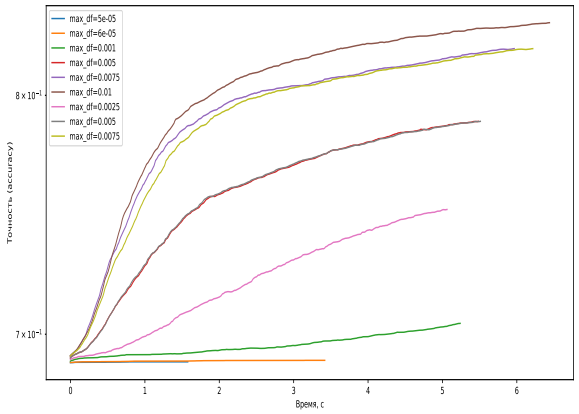
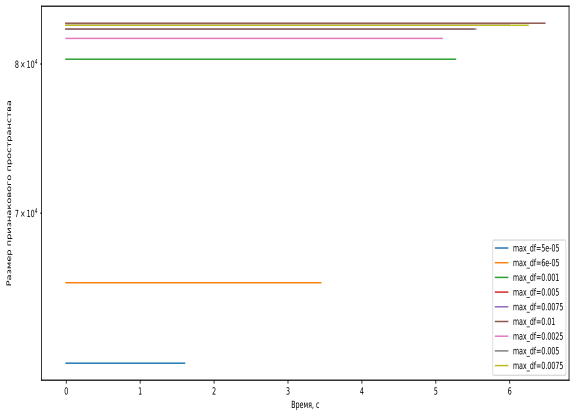


Рис. 36: Зависимость размера признакового пространства от параметра конструктора max_df



Из рис. 31 видно, что:

- при использовании представления BagOfWords достигается наибольшее значение точности (это может быть связано с тем, что токсичные комментарии более заспамленные, поэтому TF-IDF занижает вклад «спам-слов» в таких сообщениях)
- время работы метода не зависит от использования представлений BagOfWords или TF-IDF (является следствием следующего пункта)
- размерность признакового пространства не зависит от типа представления (так как в BagOfWords и TF-IDF различаются только значения признаков, но не их количество)

Из рис. 32 видно, что:

- при достаточно маленьких значениях **min_df** (до 0.0025) достигается практически одинаковая точность, но если брать значения больше 0.0025, то точность классификации ухудшается.

Из рис. 33 можно заметить, что:

- с увеличением значения **min_df** уменьшается время работы алгоритма (следствие из пункта ниже)

- с увеличением значения **min_df** уменьшается размерность признакового пространства

Из рис. 34 видно, что:

- при использовании представления BagOfWords достигается наибольшее значение точности (это может быть связано с тем, что токсичные комментарии более заспамленные, поэтому TF-IDF занижает вклад «спам-слов» в таких сообщениях)
- время работы метода не зависит от использования представлений BagOfWords или TF-IDF (является следствием следующего пункта)
- размерность признакового пространства не зависит от типа представления (так как в BagOfWords и TF-IDF различаются только значения признаков, но не их количество)

Из рис. 35 видно, что:

- с увеличением **max_df** точность классификации повышается.

Из рис. 36 можно заметить, что:

- с увеличением значения **max_df** увеличивается время работы алгоритма (следствие из пункта ниже)
- с увеличением значения **max_df** увеличивается размерность признакового пространства

4 Применение лучшего алгоритма к тестовой выборке и анализ ошибок, допущенных алгоритмом

Применим к тестовой выборке лучший алгоритм[1] Получено значение точности: **0.8854**
Проанализируем ошибки алгоритма:

- «i **didn t** screw up i lapsed when you revert you should examine what the previous editor did as several reverters are doing reverting again me at amin al husseini so blindly they do **not** notice they are removing intermediate edits involving correction of dates and spellings it means that this is personal or ideological and **not** motivated by intelligent assessments of the merits».

  Данный комментарий был классифицирован, как токсичный. Есть предположение, что алгоритм ошибается в тех комментариях, где есть много отрицаний (чаще такие комменатарии имеют негативный оттенок).

  Также алгоритм ошибается, если комментарий содержит слова с ошибками; перестановками, дублированием букв (к сожалению, цензура не позволяет приложить соответствующие примеры)

5 Бонусная часть

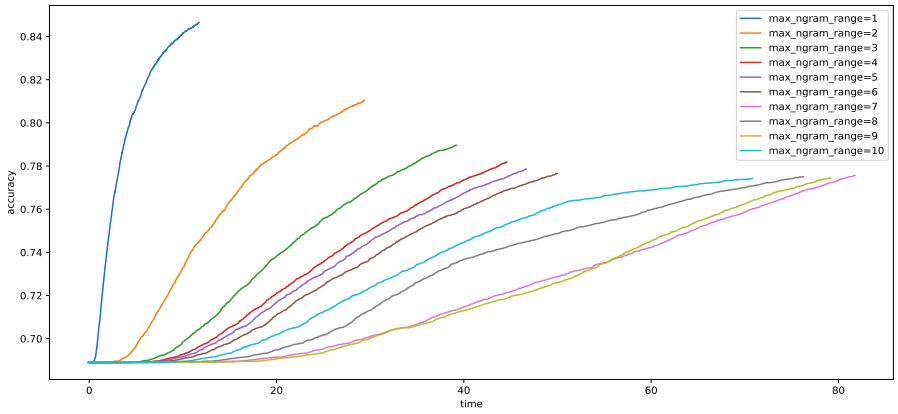
5.1 Исследование параметра ngramm_range у TfidfTransformer

Данный параметр отвечает за количество n-грамм, которые будут добавлены в признаковое пространство. n-грамма - некоторая последовательность из n элементов (в данном случае элементы - слова).

Результаты экспериментов представлены на графике ниже (рис. 37)

[1] SGD, $step\_alpha = 1$, $step\_beta = 0.001$, $batch\_size = 1024$ с преобразованием BOW, лемматизацией, но без удаления стоп-слов.

Рис. 37: Зависимость точности значения параметра **ngramm_range**



Можно заметить, что с увеличением значения параметра **ngramm_range** увеличивается и время работы алгоритма.

5.2 Улучшение качества работы линейных алгоритмов

5.2.1 Adversarial validation

Рассмотрим альтернативный способ валидации. Основная идея заключается в следующем: нужно добиться того, чтобы алгоритм не смог отличать тренировочную выборку от тестовой.

Для реализации этого нужно сформировать новую выборку $X = X_{train} \bigcup X_{test}$, где X_{train}, X_{test} – объекты тренировочной и тестовой выборок соответственно (без меток классов). Поставим каждому объекту из X_{train} в соответствие метку «0», а каждому из X_{test} – «1». Таким образом получена новая тренировочная выборка, к которой можно применить метод кросс-валидации для оценки модели.

Выбор лучших параметров происходит на основе предсказаний модели для каждого фолда. За метрику качества удобнее всего брать площадь под кривой ошибок (**AUC ROC**).

Пусть даны две модели: M_1 и M_2; auc_roc_1, и auc_roc_2 – значения площадей под кривой ошибок, усредненные по фолдам для каждой из моделей соответственно. Тогда модель M_1 считается лучше модели M_2, если выполнено следующее неравенство:

$$|auc\_roc_1 - 0.5| < |auc\_roc_2 - 0.5|$$

С помощью представленного способа валидации будет более результативный отбор признаков, так как он (способ) позволяет минимизировать отличия между тренировочной и тестовой выборок.

Но есть и минус у **Adversarial validation** – обобщающая способность модели, настроенной по этому методу будет ниже, чем у модели, настроенной с помощью стандартного метода кросс-валидации, так как в первом случае она (первая модель) учитывает структуру тестовой выборки. Если новая тестовая выборка имеет совсем другое распределение, то первая модель с бóльшей вероятностью выдаст качество хуже, чем вторая.

Применив данный способ валидации и поиск оптимальных параметров с помощью библиотеки **bayes_opt** удалось достигнуть качества: **0.8918**. Оптимальные параметры:

- step_alpha = 0.965
- step_beta = 0
- batch_size = 698
- tolerance = 1e-7
- l2_coef = 0

- max_iter = 10000

Так же были предприняты попытки усреднять предсказания модели при кросс-валидации (то есть обучив для одного фолда сразу предсказывать для теста и сохранять эти предсказания, а затем их усреднять), но это дало результат хуже.