Отчет по заданию №2: Применение линейных моделей для определения токсичности комментария

Васильев Руслан ВМК МГУ, 317 группа

22 ноября 2020 г.

Содержание

1 Теория

В данном разделе раскрываются некоторые теоретические аспекты логистической регрессии. Обосновывается эквивалентность двух подходов: метода минимизации эмпирического риска (функции потерь для всей выборки) и вероятностной модели. Также рассматривается обобщение для задачи многоклассовой классификации. Обозначения и логика рассуждений во многом соответствуют [1]. Все векторы (например, признаков или весов) в матричных выражениях, если не указано, рассматриваются как вектор-столбцы.

1.1 Общая постановка

В стандартной постановке задачи бинарной классификации рассматривается обучающая выборка

$$\mathbb{X} = (\mathbf{x}_i, y_i)_{i=1}^{l}, \qquad \begin{array}{l} \mathbf{x}_i \in \mathbb{R}^d \text{ — объекты (их признаковые описания),} \\ y_i \in \{-1, +1\} \text{ — метки классов.} \end{array}$$

Считая, что среди признаков есть константный, в линейной модели можно представить (бинарные) ответы алгоритма следующим образом:

$$a(\mathbf{x}, \mathbf{w}) = \operatorname{sign} \langle \mathbf{x}, \mathbf{w} \rangle, \quad \mathbf{w} \in \mathbb{R}^d \text{ — вектор весов.}$$

Вместо пороговой фунции потерь $\mathcal{L}_{\mathbb{1}}$ мы используем верхнюю оценку — гладкую аппроксимацию $\mathcal{L}$:

$$\mathcal{L}_{\mathbb{1}}(y, a) = [a \cdot y < 0] = [\langle \mathbf{x}, \mathbf{w} \rangle y < 0] \leqslant \mathcal{L}(\underbrace{\langle \mathbf{x}, \mathbf{w} \rangle y}_{M(\mathbf{w}) \text{ — отступ на объекте } \mathbf{x}}) \leqslant \mathcal{L}(M), \tag{1}$$

причем процесс обучения представляет собой поиск оптимального $\mathbf{w}$, минимизирующего эмпирический риск:

$$Q(\mathbb{X}, \mathbf{w}) = \frac{1}{l} \sum_{i=1}^{l} [\underbrace{\langle \mathbf{x}_i, \mathbf{w} \rangle y_i}_{M_i(\mathbf{w})} < 0] \leqslant \frac{1}{l} \sum_{i=1}^{l} \mathcal{L}(M_i(\mathbf{w})) \to \min_{\mathbf{w}}. \tag{2}$$

Для решения задачи обычно используются градиентные методы, требующие вычисление градиента $\frac{\partial \mathcal{L}}{\partial \mathbf{w}}$. Также к минимизируемому функционалу (2) могут добавляться дополнительные слагаемые-регуляризаторы. Например, в данной работе используется L_2-регуляризатор, с которым задача имеет вид:

$$\widetilde{Q}(\mathbb{X}, \mathbf{w}) = \frac{1}{l} \sum_{i=1}^{l} \mathcal{L}(M_i(\mathbf{w})) + \frac{\lambda}{2} \|\mathbf{w}\|^2 \to \min_{\mathbf{w}}. \tag{3}$$

Градиент квадратичного регуляризатора вычисляется тривиально:

$$\frac{\partial}{\partial \mathbf{w}} \left(\frac{\lambda}{2} \|\mathbf{w}\|^2 \right) = \frac{\lambda}{2} \cdot 2\mathbf{w} = \lambda \mathbf{w}. \tag{4}$$

1.2 Бинарная логистическая регрессия

1.2.1 С точки зрения минимизации эмпирического риска

Рассмотрим следующую функцию потерь[1] — ее обычно называют логарифмической или логистической:

$$\mathcal{L}(M) = \log\left(1 + e^{-M}\right) = \log\left(1 + e^{-\langle \mathbf{x},\mathbf{w}\rangle y}\right), \tag{5}$$

частная производная которой

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mathbf{w}} &= \frac{\partial}{\partial \mathbf{w}} \log\left(1 + e^{-\langle \mathbf{x},\mathbf{w}\rangle y}\right) \\
&= \frac{1}{1 + e^{-\langle \mathbf{x},\mathbf{w}\rangle y}} \cdot \frac{\partial}{\partial \mathbf{w}}\left(1 + e^{-\langle \mathbf{x},\mathbf{w}\rangle y}\right) \\
&= \frac{e^{-\langle \mathbf{x},\mathbf{w}\rangle y}}{1 + e^{-\langle \mathbf{x},\mathbf{w}\rangle y}} \cdot \frac{\partial}{\partial \mathbf{w}}\left(-\langle \mathbf{x}, \mathbf{w}\rangle\, y\right) \\
&= \frac{1}{1 + e^{\langle \mathbf{x},\mathbf{w}\rangle y}} \cdot (-y\mathbf{x}) \\
&= -y\sigma\left(-\langle \mathbf{x}, \mathbf{w}\rangle\, y\right) \cdot \mathbf{x},
\end{aligned} \tag{6}$$

где с помощью σ мы обозначили сигмоиду:

$$\sigma(s) = \frac{1}{1 + e^{-s}}.$$

Таким образом, на всей выборке имеем:

$$\frac{\partial \widetilde{Q}}{\partial \mathbf{w}} = \frac{\partial Q}{\partial \mathbf{w}} + \lambda \mathbf{w} = -\frac{1}{l}\sum_{i=1}^{l} y_i \sigma\left(-\langle \mathbf{x}_i, \mathbf{w}\rangle\, y_i\right) \cdot \mathbf{x}_i + \lambda \mathbf{w}. \tag{7}$$

1.2.2 С точки зрения вероятностной модели

Возьмем следующую модель условной вероятности:

$$y \mid \mathbf{x}, \mathbf{w} \sim \mathrm{Be}\left\{\sigma\left(\langle \mathbf{x}, \mathbf{w}\rangle\right)\right\}, \tag{8}$$

то есть $y \in \{0, 1\}$ качестве ответов алгоритма будем брать небинаризованные вероятности:

$$a(\mathbf{x}) = \mathbb{P}(y = 1 | \mathbf{x}, \mathbf{w}) = \sigma\left(\langle \mathbf{x}, \mathbf{w}\rangle\right) \tag{9}$$

Можно искать параметр $\mathbf{w}$ методом максимального правдоподобия. Это эквивалентно минимизации логарифма правдоподобия выборки, взятого со знаком минус (a_i — ответ на i-м объекте):

$$-l(\mathbf{w}) = -\log \prod_{i=1}^{l} \mathbb{P}(y_i | \mathbf{x}_i, \mathbf{w}) = -\log \prod_{i=1}^{l} a_i^{y_i}(1 - a_i)^{1-y_i} =$$

$$= \sum_{i=1}^{l}(-y_i \log a_i - (1 - y_i)\log(1 - a_i)) \;\to\; \min_{\mathbf{w}}$$

[1] Заметим, что для выполнения неравенства (1) основание логарифма должно быть в пределах $(1, 2]$. Но в силу свойств логарифма выбор основания будет влиять исключительно на масштаб, поэтому для удобства обычно бурут e.

На одном объекте ошибка имеет вид:

$$\text{logloss}\,(a, y) = -y \log a - (1 - y) \log(1 - a) \tag{10}$$

И, (10) тоже называется логистической функцией потерь. Легко показать, что (5) и (10) с подстановкой (9) — одна и та же функция потерь с точностью до переобозначения классов $\{-1, +1\} \leftrightarrow \{0, 1\}$:

$$\begin{aligned}
&- [y = +1] \log a - (1 - [y = +1]) \log(1 - a) \\
&= - [y = +1] \log \sigma\left(\langle \mathbf{x}, \mathbf{w}\rangle\right) - (1 - [y = +1]) \log(1 - \sigma\left(\langle \mathbf{x}, \mathbf{w}\rangle\right)) \\
&= - [y = +1] \log \sigma\left(\langle \mathbf{x}, \mathbf{w}\rangle\right) - [y = -1] \log \sigma\left(-\langle \mathbf{x}, \mathbf{w}\rangle\right) \\
&= - \log \sigma\left(\langle \mathbf{x}, \mathbf{w}\rangle\, y\right) \\
&= \log\left(1 + e^{-\langle \mathbf{x}, \mathbf{w}\rangle y}\right) = \mathcal{L}(M).
\end{aligned}$$

Таким образом, мы показали, что вероятностный подход эквивалентен минимизации эмпирического риска. Вероятностный смысл можно придать и регуляризации, веса тоже рассматривать как случайную величину.

Напоследок перепишем в еще одной форме градиент функции потерь (6) на одном объекте (при переименовании классов в $\{0, 1\}$, с учеотом небинаризованных ответов (9)):

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mathbf{w}} &= -y \sigma\left(-\langle \mathbf{x}, \mathbf{w}\rangle\right) \cdot \mathbf{x} + (1 - y)\, \sigma\left(\langle \mathbf{x}, \mathbf{w}\rangle\right) \cdot \mathbf{x} \\
&= -y \cdot (1 - a) \cdot \mathbf{x} + (1 - y) \cdot a \cdot \mathbf{x} \\
&= (a - y)\, \mathbf{x}.
\end{aligned} \tag{11}$$

1.3 Многоклассовая логистическая регрессия

Пусть у нас теперь не 2 класса, а m классов: $1, \ldots, m$. Начнем с вероятностной модели:

$$\mathbb{P}(y = k \mid \mathbf{x}, \mathbf{w}_1, \ldots, \mathbf{w}_m) = \text{Softmax}_k\left(\langle \mathbf{x}, \mathbf{w}_1\rangle, \ldots, \langle \mathbf{x}, \mathbf{w}_m\rangle\right), \tag{12}$$

где $\text{Softmax}\colon \mathbb{R}^m \to (0, 1)^m$ вычисляется по формуле

$$\text{Softmax}(\mathbf{s}) = \left(\frac{e^{s_1}}{\sum_{t=1}^m e^{s_t}}, \ldots, \frac{e^{s_m}}{\sum_{t=1}^m e^{s_t}}\right)$$

Достаточно уметь предсказывать только $k - 1$ класс, вероятность последнего будет получаться из нормировки. И модель действительно избыточна: ответы не изменятся при сдвиге всех весов. Чтобы устранить неоднозначность, положим $\mathbf{w}_m := \mathbf{0}$.

Теперь отдельно рассмотрим случай $m = 2$. Вектор вероятностей классов будет иметь вид:

$$\begin{aligned}
\left(\frac{e^{\langle \mathbf{x}, \mathbf{w}_1\rangle}}{e^{\langle \mathbf{x}, \mathbf{w}_1\rangle} + 1}, \frac{1}{e^{\langle \mathbf{x}, \mathbf{w}_1\rangle} + 1}\right) &= \left(\sigma\left(\langle \mathbf{x}, \mathbf{w}_1\rangle\right), \sigma\left(-\langle \mathbf{x}, \mathbf{w}_1\rangle\right)\right) = \\
&= \{\mathbf{w} := -\mathbf{w}_1\} = \left(1 - \sigma\left(\langle \mathbf{x}, \mathbf{w}\rangle\right), \sigma\left(\langle \mathbf{x}, \mathbf{w}\rangle\right)\right)
\end{aligned}$$

Мы показали, что при $m = 2$ параметрическая модель многоклассовой логистической регрессии эквивалента бинарной модели (8). И оценку весов $\mathbf{w}_1, \ldots, \mathbf{w}_{m-1}$ в

случае произвольного $m > 1$ тоже можно искать с помощью метода максимального правдоподобия. Для удобства обозначим:

$$y_{ik} = [y_i = k],$$
$$p_{ik} = \frac{e^{\langle \mathbf{x}_i, \mathbf{w}_k \rangle}}{\sum_{t=1}^{m} e^{\langle \mathbf{x}_i, \mathbf{w}_t \rangle}},$$

где $i \in \{1, \ldots, l\}$, $k \in \{1, \ldots, m\}$ — мы помним, что $\mathbf{w}_m \equiv \mathbf{0}$, но для удобства записи иногда будем оставлять его.

Функцию потерь можно вывести через правдоподобие:

$$\begin{aligned} -l(\mathbf{w}) &= -\log \prod_{i=1}^{l} \mathbb{P}(y_i | \mathbf{x}_i, \mathbf{w}_1, \ldots, \mathbf{w}_{m-1}) \\ &= -\log \prod_{i=1}^{l} \prod_{k=1}^{m} p_{ik}^{y_{ik}} \\ &= -\sum_{i=1}^{l} \sum_{k=1}^{m} y_{ik} \log p_{ik}, \quad \text{(разделим на длину выборки)} \\ Q(\mathbb{X}, \mathbf{w}) &= -\frac{1}{l} \sum_{i=1}^{l} \sum_{k=1}^{m} y_{ik} \log p_{ik}. \end{aligned}$$

Получили обобщение логлосса (кросс-энтропии) для нескольких классов. При решении задачи оптимизации понадобится градиент, в данном случае представляющий собой матрицу размера $d \times (m-1)$ — найдем его. Сначала распишем потери на одном объекте:

$$\begin{aligned} \mathcal{L}_i = -\sum_{k=1}^{m} y_{ik} \log p_{ik} &= -\sum_{k=1}^{m} y_{ik} \log \frac{e^{\langle \mathbf{x}_i, \mathbf{w}_k \rangle}}{\sum_{t=1}^{m} e^{\langle \mathbf{x}_i, \mathbf{w}_t \rangle}} \\ &= -\sum_{k=1}^{m} y_{ik} \langle \mathbf{x}_i, \mathbf{w}_k \rangle + \log \sum_{t=1}^{m} e^{\langle \mathbf{x}_i, \mathbf{w}_t \rangle} \\ &= \log \left(1 + \sum_{k=1}^{m-1} e^{\langle \mathbf{x}_i, \mathbf{w}_k \rangle}\right) - \sum_{k=1}^{m-1} y_{ik} \langle \mathbf{x}_i, \mathbf{w}_k \rangle \end{aligned}$$

Найдем ее градиент по вектору весов $\mathbf{w}_j$:

$$\frac{\partial \mathcal{L}_i}{\partial \mathbf{w}_j} = \frac{1}{1 + \sum_{k=1}^{m-1} e^{\langle \mathbf{x}_i, \mathbf{w}_k \rangle}} \cdot e^{\langle \mathbf{x}_i, \mathbf{w}_j \rangle} \cdot \mathbf{x}_i - y_{ik} \mathbf{x}_i = \left(p_{ij} - y_{ij}\right) \mathbf{x}_i. \tag{13}$$

Выражение (13) совпало с (11)!

Тогда градиент полной функции потерь по произвольному вектору весов:

$$\frac{\partial Q}{\partial \mathbf{w}_j} = \frac{1}{l} \sum_{i=1}^{l} \left(p_{ij} - y_{ij}\right) \cdot \mathbf{x}_i.$$

Объединяя в матрицу градиенты Q по $\mathbf{w}_1, \ldots, \mathbf{w}_{m-1}$ (то есть конкатенируя столбцы), получаем полную производную функции потерь.

2 Эксперименты

2.1 Структура

В задании предлагаются обучающая и тестовая выборка, составленные на основе соревнования [2]. Необходимо решить задачу бинарной классификации, реализовав логистическую регрессию, а также оптимизационные алгоритмы: градиентный спуск и стохастический градиентный спуск. В ходе экспериментов рассматривается множество гиперпараметров, в частности зависимость от времени и итераций, а также варианты предобработки текстовых данных.

Параметры модели рассматриваются последовательно, некоторые подбираются в комбинациях друг с другом. Оставшиеся фиксируются значениями по умолчанию или соображение по предыдущим этапам эксперимента. То же относится и к предобработке данных. Тем не менее в процессе добавления этапов предобработки, подбора новых параметров, старые иногда могут оказаться нерелевантны.

Для оценки качества в задании предлагается использовать функцию потерь (усредненный логлосс с регуляризацией) и долю правильных ответов (accuracy). Чтобы более корректно оценивать алгоритм, измерения accuracy в ходе подбора гипермараметров производятся на отложенной выборке, составляющей примерно четверть исходной обучающей. Точное разбиение приведено в табл. 1. Проведя все необходимые эксперименты (для поиска параметров градиентных методов), мы обучим модель на всех исходных данных (обучающей выборки) и измерим качество на тестовой. В заданиях, связанных с предобработкой текстовых данных мы будем работать уже с полной обучающей выборкой, поскольку в задании требуется строить признаковое описание на ней.

Тип выборки	**Обучающая**	**Отложенная**	**Тестовая**
Число объектов	40000	12061	20676

Таблица 1: Разбиение по выборкам

Базовая предобработка датасета выполнена в соответствии с заданием, причем `min_df=5` (у `CountVectorizer`). Если не обговаривается, далее у функций использованы базовые параметры по умолчанию ($\alpha = 1$, $\beta = 0$, `tolerance`$=10^{-5}$, размер батча у SGD 500, параметр регуляризации $\lambda = 0$, т.к. модели оказались не склонными к переобучению).

Введен новый параметр `intercept`, соответствующий смещению или добавлению константного признака, как было описано в теоретической части, далее он всегда будет будет использоваться.

2.2 Темп обучения

В задании предлагается использовать темп обучения

$$\eta_k = \frac{\alpha}{k^\beta}, \tag{14}$$

где k — номер итерации (эпохи), α, β — константы. Поскольку α и β входят в формулу (14) одновременно, разумно подбирать их вместе. Для визуализации на каждом графике зафиксирован β и варьируется α, поскольку в теоретическое обоснование сходимости (стохастического) градиентного спуска с шагом (14) определяется именно показателем степени.

2.2.1 Градиентный спуск

Для обыкновенного градиентного спуска визуализируем зависимость функции потерь и точности от номера итерации. Группируем графики, как было обосновано выше, по значениям β.

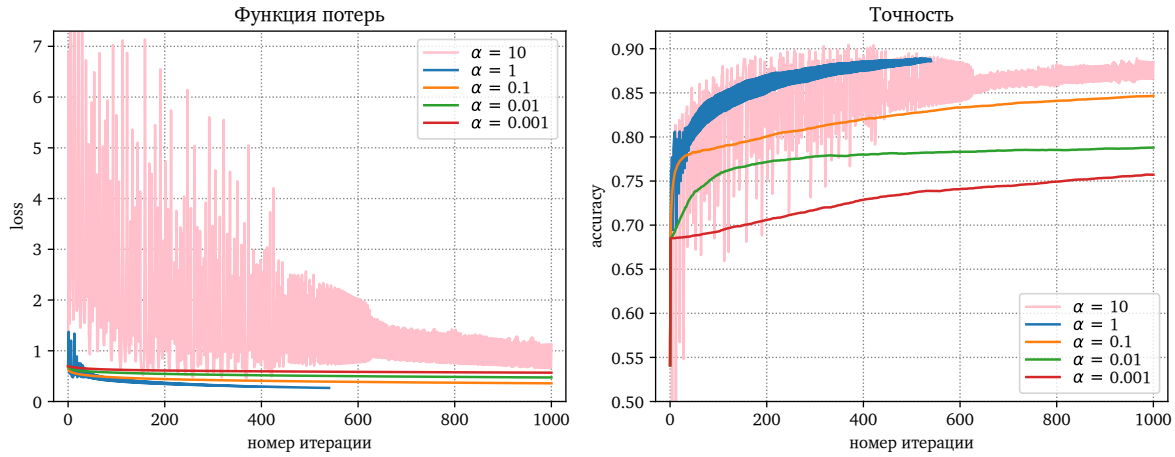

Рис. 1: Градиентный спуск, $\beta = 0$. Крайние значения при $\alpha = 10$ обрезаны.

$\beta = 0$ (рис. 1) соответствует константному шагу. При больших значениях α имеется выраженная зигзагообразность и у графика функции потерь, и у доли правильных ответов. Особенно колебания проявляются на первых итерациях. При этом можно заметить, что $\alpha = 10$, несмотря на нестабильное поведение, дает лучшее качество на некоторых итерациях! Но именно график по итерациям показывает, что за этим следует изменение в худшую сторону. При остальных константных шагах α поведение функции потерь и точности достаточно стабильное. Причем с уменьшением α алгоритм сходится медленнее.

С добавлением показателя степени $\beta = 0.1$ (рис. 2) графики немного сглаживаются, а также можно видеть снижение разброса с ростом итерации. При $\alpha = 10$ сходимость функции потерь ускорилась, но при малых значениях α оба графика быстрее выходят на асимптоту.

При $\beta = \frac{1}{2}$ (рис. 3) в полной мере видна монотонная зависимость от α как потерь, так и точности: с ростом α сходимость улучшается. Хотя при заданном β тысячи итераций еще недостаточно для удовлетворения условия сходимости, графики уже становятся пологими, и имеет смысл настраивать число итераций только на больших α.

Когда мы возьмем $\beta = 1$ (рис. 4), градиентный спуск сходится достаточно быстро. Внутри алгоритма под «сходмостью» понимается выравнивание функции потерь, то есть отсутствие выраженных изменений на соседних итерациях. И при

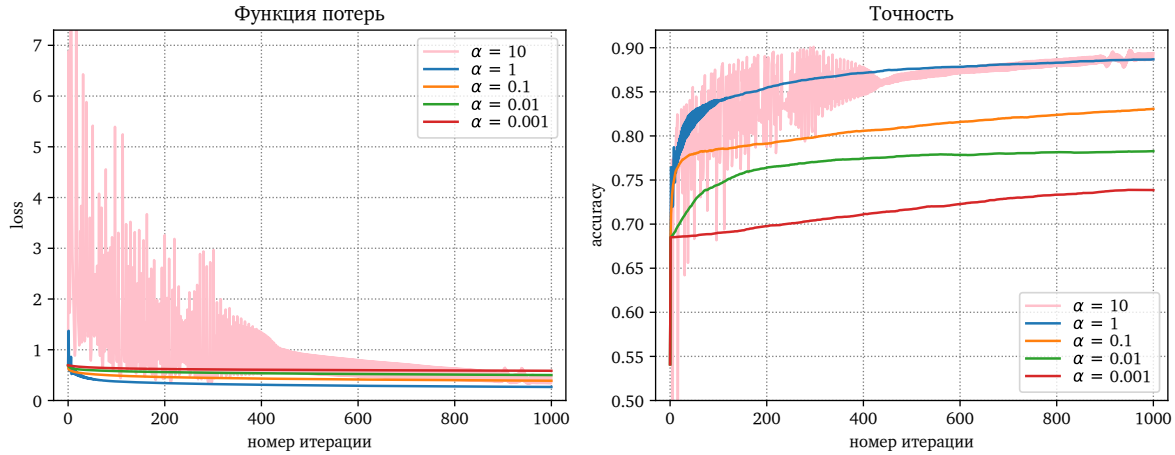

Рис. 2: Градиентный спуск, $\beta = 0.1$. Крайние значения при $\alpha = 10$ обрезаны.

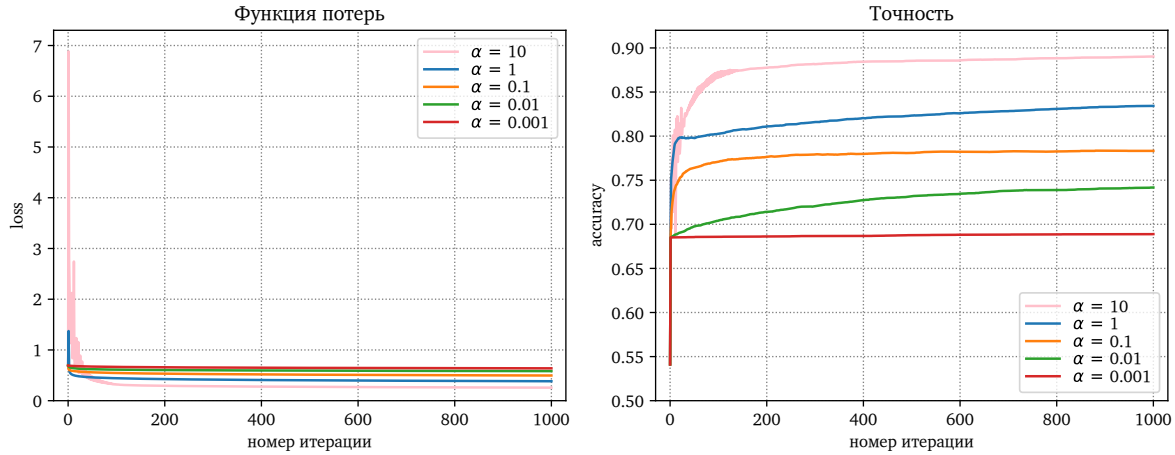

Рис. 3: Градиентный спуск, $\beta = \frac{1}{2}$.

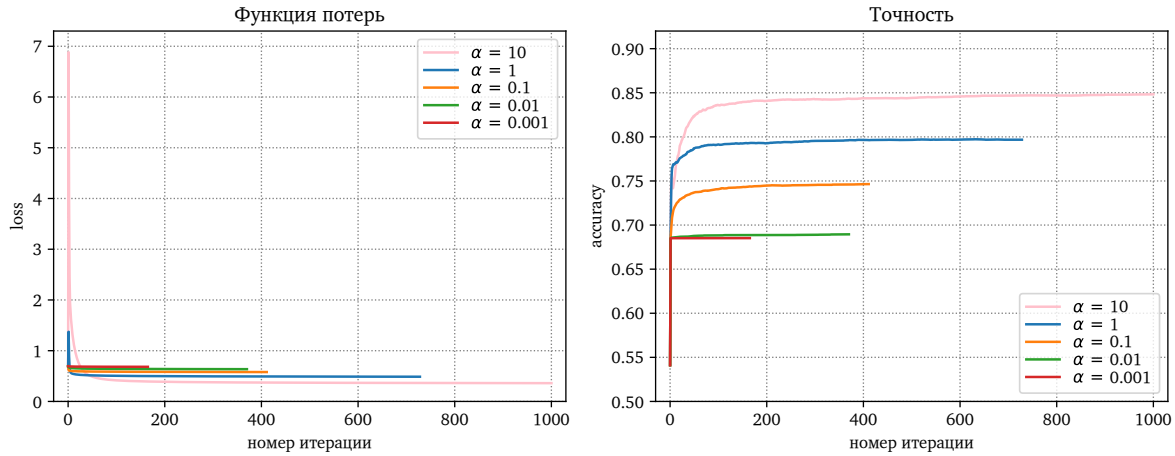

Рис. 4: Градиентный спуск, $\beta = 1$.

$\beta = 1$ проявляется недостаток такого критерия останова: по точности видно, что

веса не достигают оптимальных значений.

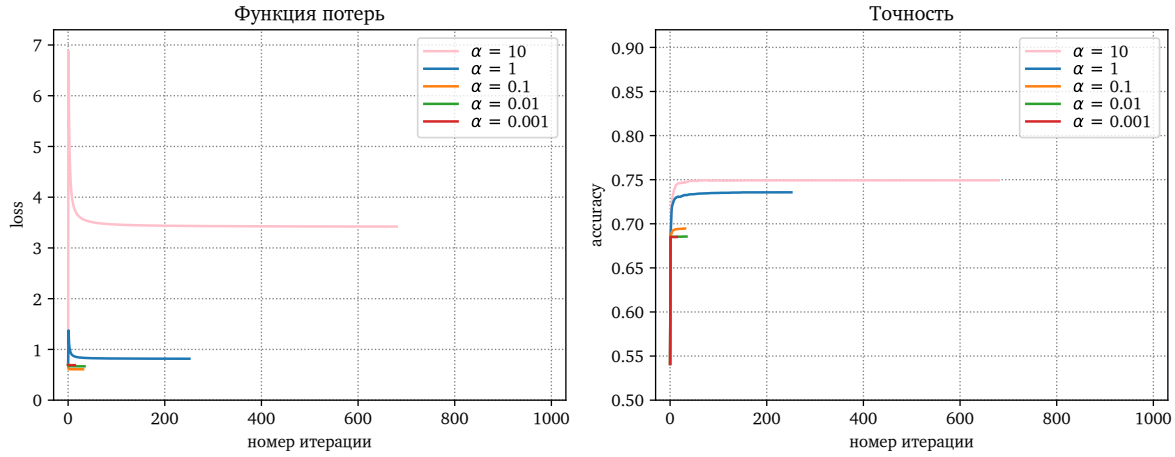

Рис. 5: Градиентный спуск, $\beta = 2$.

Максимальное исследуемое $\beta = 2$ (рис. 4) только усиливает проблемы, возникшие при $\beta = 1$. Алгоритмы сходятся очень быстро, но результирующее качество крайне низкое. Однако сохраняется монотонная зависимость от α.

Итак, при больших значениях α можно получить наилучшее качество, но поведение вектора весов становится нестабильным. Уменьшать этот эффект можно с помощью увеличения параметра β. Но рост β приводит к быстрому выходу функции потерь на асимптоту, и веса не успевают сойтись к нужным значениям, обеспечивающим высокое качество.

Неплохо проявили себя две стратегии. Первая (и простая по настройке) — работать с константным шагом ($\beta = 0$), но при этом нужно брать $\alpha \approx 1$, т.к. при чрезмерно больших значениях пропадет стабильность, а при меньших — замедляется сходимость и падает качество. Вторая — брать малое $\beta > 0$ (в данном случае $0.1-0.5$) и тогда достаточно большие $\alpha \geqslant 1$ могут дать не менее хороший результат.

2.2.2 Стохастический градиентный спуск

Теперь рассмотрим, как повлияют параметры α и β на поведение стохастического градиентного спуска. Дизайн эксперимента совпадает с градиентым спуском. Только в данном случае вместо номера итерации используется, как предложено в задании, приближенный номер эпохи. Под данной величиной понимается Доля объектов, на которых уже считался градиент, относительно размера выборки. Эта величина имеет смысл, поскольку на каждом шаге нашего варианта SGD берется не один случайной объект, а подмножество (иногда такой подход называют mini-batch gradient descent).

Функция потерь и точность обновляются тогда, когда разница между текущим и предыдущим приближенным номером эпохи > 1 (реализация допускает и более частое/редкое обновление), что в определенном смысле близко к итерации обыкновенного градиентного спуска.

На рис. 6 – рис. 10 приведены графики функции потерь и точности на тех же наборах α и β. В целом поведение очень похоже на градиентный спуск. Аналогично на

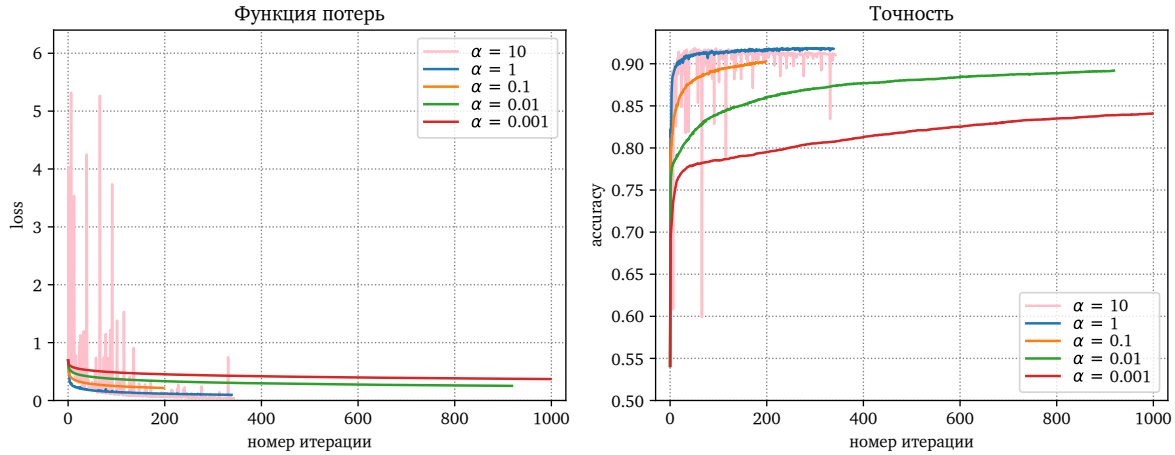

Рис. 6: Стохастический градиентный спуск, $\beta = 0$

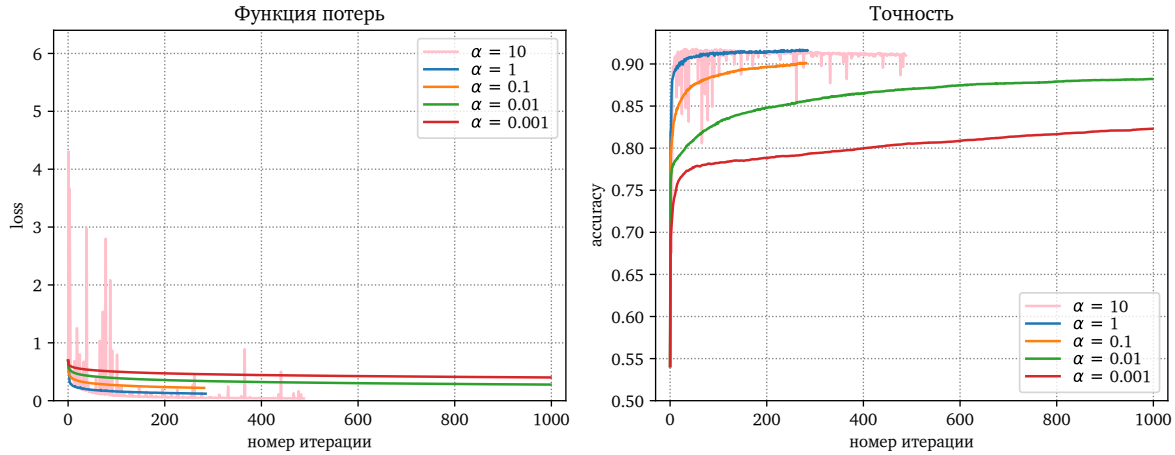

Рис. 7: Стохастический градиентный спуск, $\beta = 0.1$

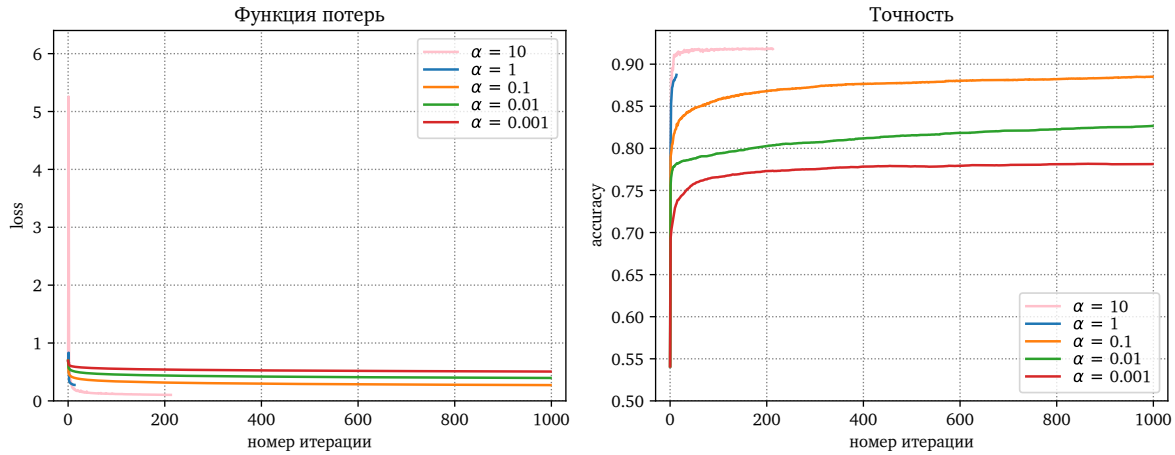

Рис. 8: Стохастический градиентный спуск, $\beta = \frac{1}{2}$

больших α достигается большее качество, рост β стабилизирует поведение графи-

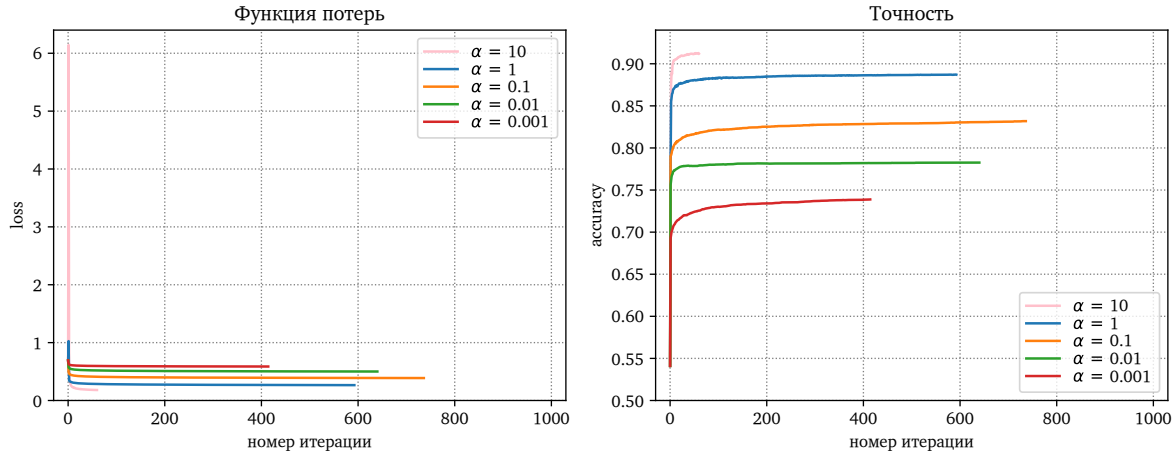

Рис. 9: Стохастический градиентный спуск, $\beta = 1$

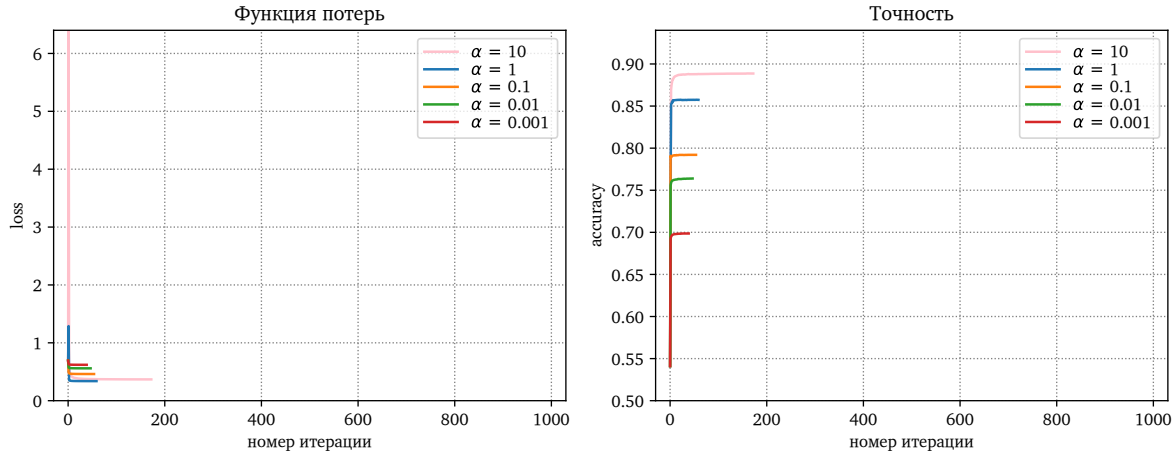

Рис. 10: Стохастический градиентный спуск, $\beta = 2$

ков. Отметим отличия: при константном шаге (рис. 6) и относительно небольшом (рис. 6) α точность выходит на асимтоту, но алгоритм при этом не останавливается. И действительно, константные шаги для стохастического градиента не гарантируют сходимость, но, например, при $\alpha = 1$, поведение все-таки приемлемое. Важно помнить, что мы учимся не на одном объекте, а на батчах. Начиная с $\beta = \frac{1}{2}$ алгоритм работает более предсказуемо: точность и потери монотонно зависят от α и большие β позволяют брать большие α. Но при слишком сильном затухании ($\beta = 2$, рис. 10) к оптимальным весам алгоритм не сходится.

Таким образом, выбор α и β для SGD в данной задаче можно проводить с теми же двумя стратекиями, что и при обычном градиентном спуске. Простая — взять $\beta = 0$ и $\alpha \approx 1$, но в данном случае нельзя особо увеличивать α. Более сложная настройка — $\beta < 2$, причем в при достаточно большом α имеет смысл и $\beta = 1$, в то время как при $\beta = 0.1$ лучше взять $\alpha \approx 1$.

2.3 Начальное приближение

Рассмотрим следующие начальные приближения для вектора весов $\mathbf{w}$:

1. $\mathbf{w}_i = 0$,
2. $\mathbf{w}_i = 1$,
3. $\mathbf{w}_i \sim U[-\frac{1}{2d}, \frac{1}{2d}]$,
4. $\mathbf{w}_i \sim U[-\frac{1}{\sqrt{d}}, \frac{1}{\sqrt{d}}]$,
5. $\mathbf{w}_i = \frac{\langle \mathbf{y}, \mathbf{x}'_i \rangle}{\langle \mathbf{x}'_i, \mathbf{x}'_i \rangle}$, где $\mathbf{y}$ — вектор меток, $\mathbf{x}'_i$ — вектор значений i-го признака.

Идея 5-й инициализации взята из [1]. Хотя для этот используется в задачах с квадратичной функцией потерь, результат окажется интересным.

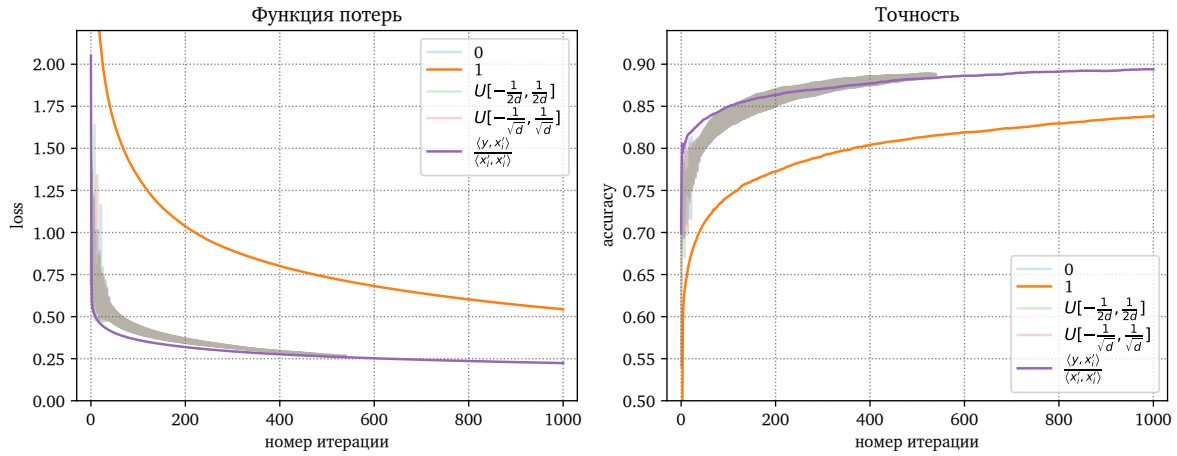

Рис. 11: Инициализация весов, градиентный спуск

На рис. 11, рис. 12 видно, что инициализация случайными значениями в окрестности 0 или непосредственно нулями приводит примерно к одинаковым результатом — у графиков специально включена прозрачность, поскольку они сливаются. Причем схожесть проявляется даже в том, что они все ведут себя достаточно зигзагообразно.

Инициализация единицами дает наихудшие результаты. В нашего признакового пространства эта оценка начинает с предсказаний константой 1 — из-за дисбаланса классов алгоритм стартует с точности, близкой к 0.3. Функция потерь тоже достаточно велика и обрезана на обоих графиках.

Куда интеревнее получилось с результатами при использовании 5-го начального приближения. В обыкновенном градиентном спуске (рис. 11) такое приближение сделало алгоритм более стабильным, чем при инициализации в окрестности нулей. В случае стохастического градиента (рис. 12) приближение дало более быструю сходимость, чем при случайной инициализации, но чуть более медленную, чем при нулевом начальном приближении. Так или иначе, разница несущественна.

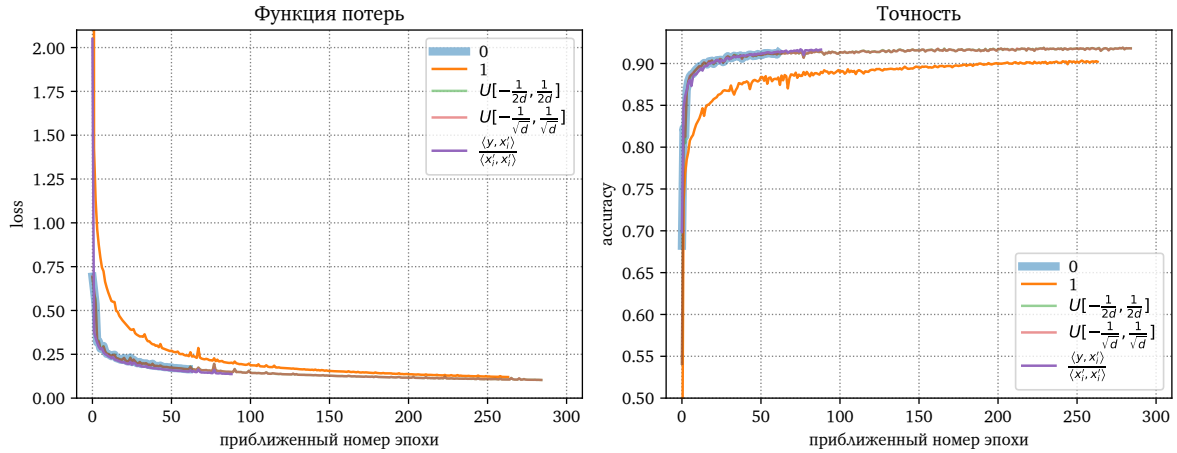

Рис. 12: Инициализация весов, стохастический градиентный спуск

Таким образом, для данной задачи можно использовать любое из приведенных начальных приближений, кроме единиц. При выборе обыкновенного градиентного спуска стоит обратить внимание на 5-й вариант. В случае стохастического градиента удобно инициализировать нулями.

2.4 Размер батча (для стохастического градиента)

Мощность подвыборки, на которой пересчитывается градиент, имеет достаточно большое влияние на качество и скорость работы алгоритма. Скорость будет рассмотрена в следующем пункте, а сейчас рассмотрим, как будут изменяться кривые в уже привычных координатах (приближенный номер эпохи $\times$ функционал). Размеры батча рассмотрим по логарифмической сетке.

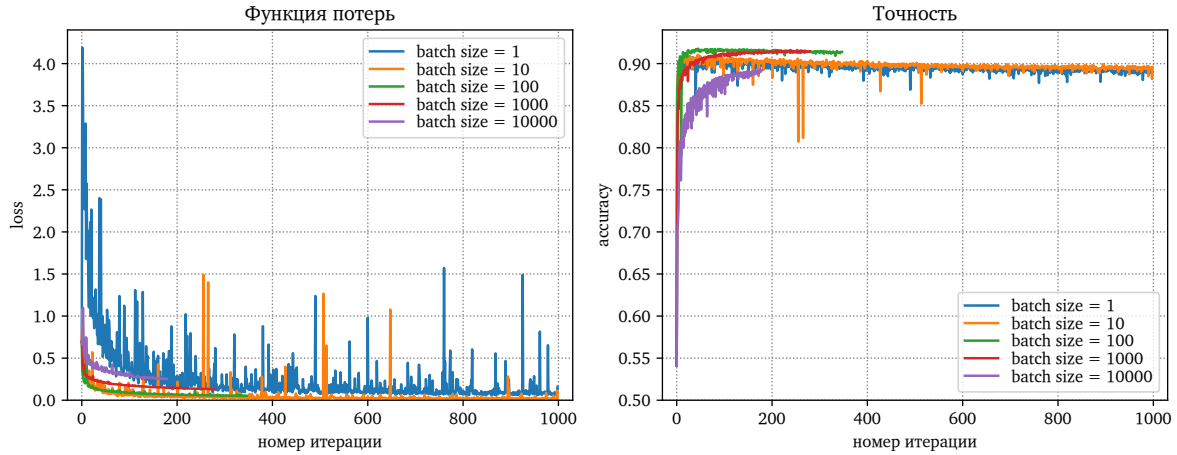

Рис. 13: Стохастический градиент, константный шаг

Поскольку с уменьшением размера батча увеличивается число шагов, важно задуматься о показателе степени в темпе обучения выр. 14. На графиках рис. 13 выбрано $\beta = 0$. При константном шаге и маленьком (1 – 10 объектов) батче алгоритм не сходится. То есть функция потерь «в целом» приближается к нулю, однако

постоянные сдвиги и относительно нечастый пересчет (фактически раз в эпоху) делают алгоритм крайне нестабильным. На правом графике видно, что точность даже падает. Добавление β сглаживает работу алгоритма (рис. 14) — в этом случае имеется сходимость и при мальеньком батче.

Когда мы берем соизмеримый с размером выборки батч (10 тыс. объектов), то алгоритм ведет себя аналогично обыкновенному градиентному спуску. Видно сходство с графиками рис. 1 и рис. 3.

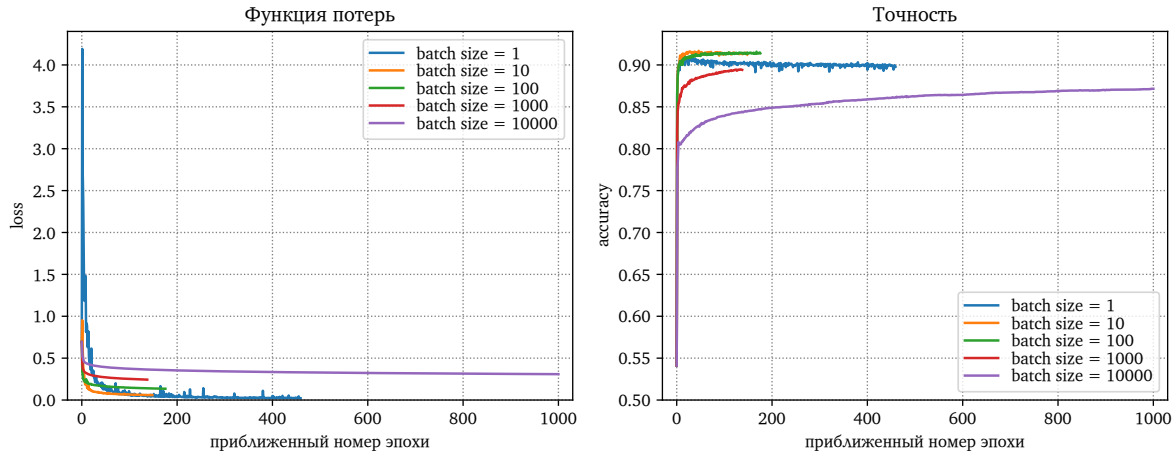

Рис. 14: Стохастический градиент, $\beta = \frac{1}{2}$

Но по графику точности становится ясно, что лучше брать размер батча 100 – 1000. В таком случае алгоритм сходится после относительно небольшого числа эпох, без выраженных зигзагов и получая высокую точность.

2.5 Время работы

Для наибольше ясности перестроим графики рис. 13 и рис. 14 относительно времени по оси абсцисс — получим рис. 15 и рис. 16.

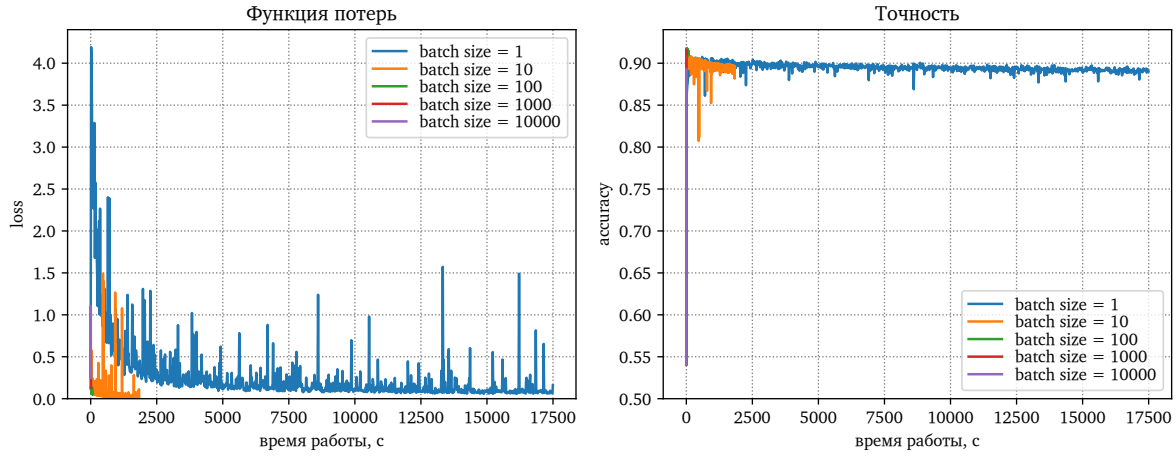

Рис. 15: Стохастический градиент, константный шаг

Если по некоторым причинам «17500 секунд» не выглядят пугающе, то это чуть меньше 5-ти часов. Ненулевое β немного спасает ситуацию (...2 часа), но все равно, вывод очевиден: в нашей задаче слишком маленькие батчи запрещены к использованию. Попробуем зафиксировать комбинации параметров, при которых и градиентный спуск, и стохастический градиентный могут показать неплохой результат.

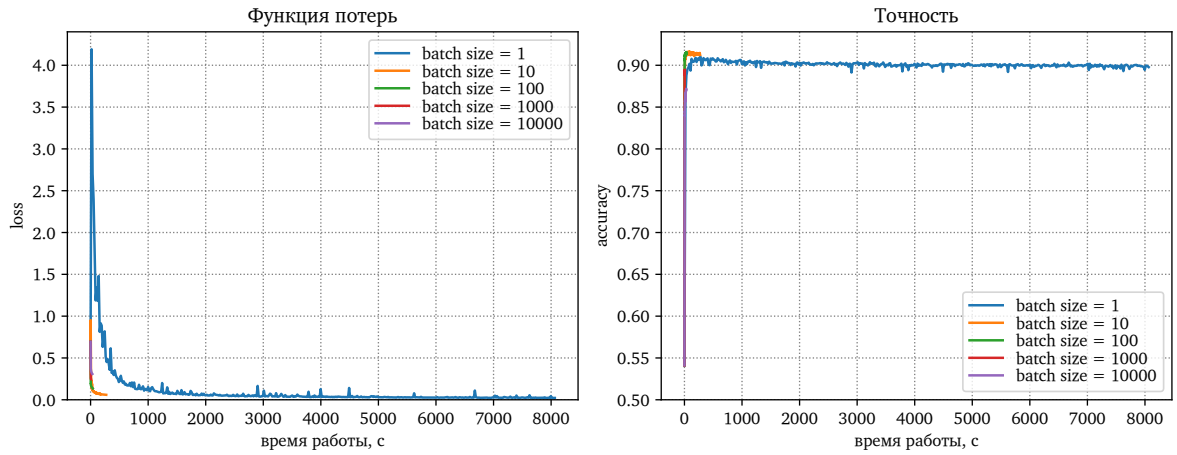

Рис. 16: Стохастический градиент, $\beta = \frac{1}{2}$

На рис. 17, рис. 18 показаны результаты. Выбирается несколько сидов для генерации начального приближения (в случае стохастического градиента и для выбора батчей). Как видим, стохастический градиент в особенности быстр при большем

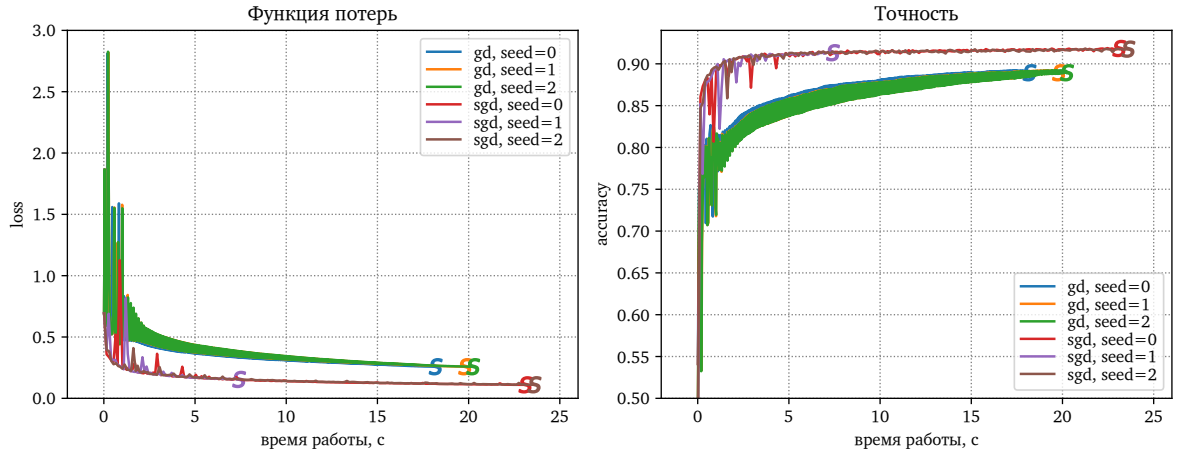

Рис. 17: Работа алгоритмов при $\alpha = 1.5,\ \beta = 0.05$. «$S$» отмечено окончание работы.

β. В случае большего масштаба α и меньшего β быстрее сходится обыкновенный градиентный спуск. При этом точность и функция потерь у случае стохастического градиента лучше.

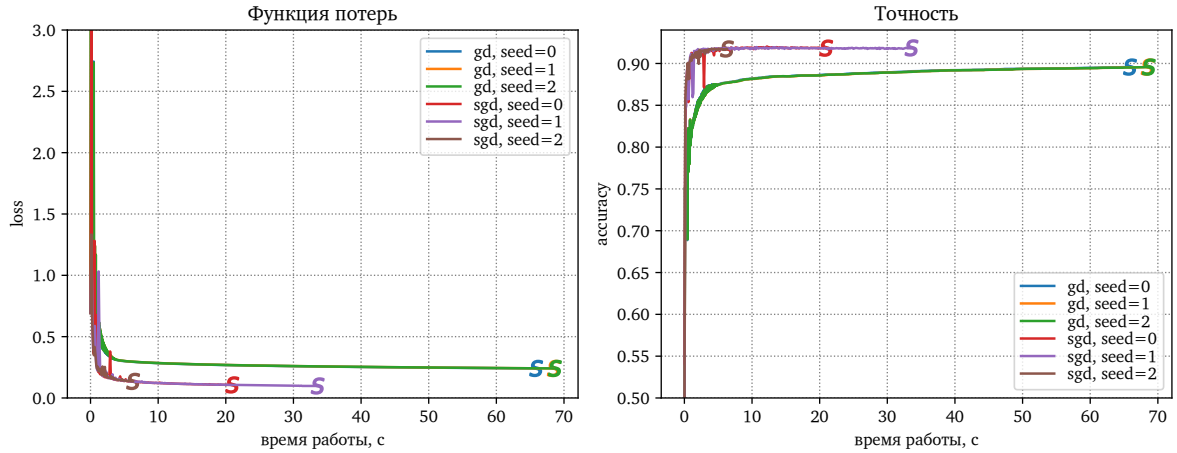

Рис. 18: Работа алгоритмов при $\alpha = 10,\ \beta = \frac{3}{2}$. «$S$» отмечено окончание работы.

2.6 Промежуточные итоги по параметрам градиентных методов

Итак, на данный момент мы провели анализ параметров стохастического градиентного спуска и сопоставили их друг с другом. Из результатов очевидно, что стохастический градиент предпочтительнее. На валидационной выборке мы попробовали разные комбинации — далее по умолчанию возьмем $\alpha = 1.5$, $\beta = 0.05$, начальное приближение из $\sim U[-\frac{1}{2d}, \frac{1}{2d}]$. Обучим модель на полной обучающей выборке. В результате получим точность на тестовой: 0.8876.

2.7 Лемматизация и удаление стоп-слов

Поробуем, помимо базовой предобработки, применить лемматизацию и удаление стоп-слов к нашему корпусу. Результаты приедены в табл. 2.

лемматизация	-	+	-	+
удаление стоп-слов	-	-	+	+
число признаков	18253	16374	18109	16234
время работы, с	46	56	33	23
точность	0.8878	0.8878	0.8859	0.8870

Таблица 2: Предобработка текста

Из результатов видим, что лемматизация в значительной мере уменьшает размерность признакового пространства, но не понижает качество. В то время как удаление стоп-слов ухудшает качество модели, несмотря на то что алгоритм быстрее заканчивает работу.

Далее оставляем лемматизацию.

2.8 Настройка числовых признаков

Теперь посмотрим, как повлияют на качество, скорость работы и размерность признакового пространства параметры числовых моделей — Bag of words и TFIDF.

min_df	1	3	5	10	20	30	40	45
число признаков	82991	23925	16374	10210	6646	5163	4298	3982
время работы, с	41 88	39 39	57 38	39 37	10 7	14 14	10 14	34 12
точность	0.8902 0.8929	0.8875 0.8916	0.8878 0.8916	0.8896 0.8915	**0.8912** 0.8894	0.8849 **0.8934**	0.8881 0.8925	0.8837 0.8907

Таблица 3: Варьирование `min_df`, в сдвоенных результатах сверху BoW, снизу TFIDF

Будем сначала удалять слишком редкие слова, увеличивая значение `min_df`. Результаты приведены d табл. 3. Число признаков очень быстро падает с ростом `min_df` — здесь все очевидно. А вот у точности нет такой монотонности. Модель bag of words почти не показывает закономерностей в данном параметре, а вот точность TFIDF, по-видимому, достигает максимума в окрестности 30. Возьмем, например, `min_df`=25 и посмотрим как повлияет выбрасывание самых частых слов коллекции, за что отвечает параметр `max_df`.

max_df	1	0.3	0.15	0.1	0.05
число признаков	5801	5790	5765	5747	5690
время работы, с	8 7	2 13	3 14	12 16	6 16
точность	0.8890 0.8903	0.8885 0.8926	**0.8890** 0.8918	0.8856 **0.8934**	0.8870 0.8929

Таблица 4: Подбор `max_df`, в сдвоенных результатах сверху BoW, снизу TFIDF

Отметим, что TFIDF во всех случях дает лучшее качество, чем BoW. Время работы алгоритмов после фиксации числа признаков почти не зависит от параметров, а скорее случайно.

Причем мы помним, что выбрасывание стоп-слов ухудшило модель. Почему тогда здесь увеличение `max_df` не снижает качество? Все очень просто: stop-words включает в себя в целом распространенные слова, в то время как `max_df` удаляет высокочастотные объекты для данной коллекции. Например, в терминах сегодняшнего твиттера в роли таких слов могли бы выступить слова из Trends, на которые комментарии бывают и позитивными, и негативными.

2.9 n-граммы

Добавление n-грамм не улучшает качество и время в данной задаче, что сразу видно в табл. 5.

ngram_range	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)
число признаков	5747	18034	23092	24914	26288
время работы, с	18	43	58	67	66
точность	**0.8934**	0.8923	0.8926	0.8929	0.8926

Таблица 5: Добавление n-грамм

2.10 Заключение

Таким образом, наилучшая модель ($\alpha = 1.5$, $\beta = 0.05$, `min_df=25`, `max_df=0.1`, использование TFIDF; остальное — по умолчанию) дает на тестовой выборке качество 0.8934. Рассмотрим несколько ошибок, указывая в скобках тип: класс P — токсичный комментарий, N — не токсичный. Поскольку почти во всех комментариях присутствует ненормативная лексика, попробуем найти не самые страшные:

1. `':Dear god this site is horrible.'` (FP) Данный комментарий вне контекста вполне относится к классу P.

358. `'666 the devil will get you all !!!!!!!!!!!!!!! !!!!!!!!!!!!!!'` (FP) То же справедливо и здесь. Судя по остальным комментариям, основные ошибки алгоритма именно на тех объектах, которые действительно могут быть отнесены к положительному классу. И здесь проявляется ситуация, когда знаки препинания также содержат информациюю для классификации.

6483. `'==The article was locked!== I've got an account, but it is annoying how some silly dimwit caused this to get locked!'` (FN) В данном случае сообщение выражает негодование человека в связи с отсутствием доступа к статье. Слова article, account, locked обычно не распространены в токсичных комментариях.

8528. `'You are a gimp'` (FN) Слово «gimp» сравнительно редко используется.

Список литературы

[1] К. В. Воронцов. *Машинное обучение (курс лекций)*. URL: machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5_(%D0%BA%D1%83%D1%80%D1%81_%D0%BB%D0%B5%D0%BA%D1%86%D0%B8%D0%B9,_%D0%9A.%D0%92.%D0%92%D0%BE%D1%80%D0%BE%D0%BD%D1%86%D0%BE%D0%B2).

[2] *Toxic Comment Classification Challenge*. URL: kaggle.com/c/jigsaw-toxic-comment-classification-challenge.