

Исследование работы линейных моделей

Тыцкий Владислав

Ноябрь 2020

Градиент в логистической регрессии

Пусть $X \in R^{N \times F}$ – матрица объектов-признаков, $y \in \{-1, 1\}^N$ – метки соответствующих объектов, $w \in R^F$ – вектор весов, x_i , – i -ый объект y_i – метка класса i -ого объекта, T – стохастическая подвыборка ($|T|$ и T будут обозначаться T в зависимости от контекста). Везде будем считать, что добавлен константный признак.

Дана задача оптимизации:

$$Q(X, y, w) = \mathcal{L}(X, y, w) + \frac{\lambda}{2} \|w\|_2^2 \rightarrow \min_w$$
$$\mathcal{L}(X, y, w) = \frac{1}{T} \sum_{i=1}^T \log(1 + \exp(-y_i \langle x_i, w \rangle)), T \leq N$$

Для решения этой задачи с помощью градиентных методов необходимо знать градиент функционала $Q(X, y, w)$

$$dQ = d\mathcal{L} + \frac{\lambda}{2} d\langle w, w \rangle = d\mathcal{L} + \lambda \langle w, dw \rangle$$
$$d\mathcal{L} = \frac{1}{T} \sum_{i=1}^T \frac{d(\exp(-y_i \langle x_i, w \rangle))}{1 + \exp(-y_i \langle x_i, w \rangle)} = \frac{1}{T} \sum_{i=1}^T \frac{\exp(-y_i \langle x_i, w \rangle) d\langle -y_i x_i, w \rangle}{1 + \exp(-y_i \langle x_i, w \rangle)} =$$
$$- \frac{1}{T} \sum_{i=1}^T \frac{\langle y_i x_i, dw \rangle}{1 + \exp(y_i \langle x_i, w \rangle)}$$

Заметим, что $dQ = \langle \nabla Q, dw \rangle$. Окончательно получаем:

$$\nabla Q(X, y, w) = \lambda w - \frac{1}{T} \sum_{i=1}^T \frac{y_i x_i}{1 + \exp(y_i \langle x_i, w \rangle)}$$

Случай для K классов

Пусть $X \in R^{N \times F}$ – матрица объектов-признаков, $y \in K^N$ – метки соответствующих объектов, где $K = \{1 \dots k\}$ – множество классов, $w_i \in R^F$ – вектор весов

соответствующий k -ому классу, x_i – i -ый объект y_i – метка класса i -ого объекта соответственно, T – стохастическая подвыборка ($|T|$ и T будут обозначаться T в зависимости от контекста).

Дана задача оптимизации — максимизация правдоподобия:

$$Q(X, y, w) = -\frac{1}{T} \sum_{i=1}^T \log \mathbb{P}(y_i | x_i) + \frac{\lambda}{2} \sum_{k=1}^K \|w_k\|_2^2 \rightarrow \min_{w_1 \dots w_K}$$

$$\mathbb{P}(y = c | x) = \frac{\exp \langle w_c, x \rangle}{\sum_{k=1}^K \exp \langle w_k, x \rangle}$$

Найдем градиент по w_m .

$$dQ(X, y, w) = d\left(-\frac{1}{T} \sum_{i=1}^T \log \mathbb{P}(y_i | x_i)\right) + d\left(\frac{\lambda}{2} \sum_{k=1}^K \|w_k\|_2^2\right) = -\frac{1}{T} d\left(\sum_{i=1}^T \log \mathbb{P}(y_i | x_i)\right) + \lambda(w_m, dw_m)$$

$$d\left(\sum_{i=1}^T \log \mathbb{P}(y_i | x_i)\right)_{w_m} = \sum_{i=1}^T d(\log \exp \langle w_{y_i}, x_i \rangle) - \sum_{i=1}^T d\left(\log \sum_{k=1}^K \exp \langle w_k, x_i \rangle\right) =$$

$$= \sum_{\substack{i: y_i = w_m \\ i \in T}} \langle x_i, dw_m \rangle - \sum_{i=1}^T \frac{\exp \langle w_m, x_i \rangle \langle x_i, dw_m \rangle}{\sum_{k=1}^K \exp \langle w_k, x_i \rangle}$$

Отсюда получаем:

$$\nabla Q_{w_m} = \lambda w_m + \frac{1}{T} \sum_{i=1}^T \frac{\exp \langle w_m, x_i \rangle x_i}{\sum_{k=1}^K \exp \langle w_k, x_i \rangle} - \frac{1}{T} \sum_{\substack{i: y_i = w_m \\ i \in T}} x_i$$

Эквивалентность бинарной логистической регрессии и мультиномиальной при $K=2$

Доказательство.

Пусть w_+ — вектор весов соответствующий первому классу, а w_- — -1 классу.

Введем $w = w_+ - w_-$. Рассмотрим задачу мультиномиальной регрессии при $K=2$:

$$Q(X, y, w) = -\frac{1}{T} \sum_{i=1}^T \log \frac{\exp \langle w_c, x \rangle}{\sum_{k=1}^K \exp \langle w_k, x \rangle} = -\frac{1}{T} \sum_{\substack{i: y_i = w_+ \\ i \in T}} \log \frac{\exp \langle w_+, x_i \rangle}{\exp \langle w_+, x_i \rangle + \exp \langle w_-, x_i \rangle}$$

$$- \frac{1}{T} \sum_{\substack{i: y_i = w_- \\ i \in T}} \log \frac{\exp \langle w_-, x_i \rangle}{\exp \langle w_+, x_i \rangle + \exp \langle w_-, x_i \rangle} = -\frac{1}{T} \sum_{\substack{i: y_i = w_+ \\ i \in T}} \log \frac{1}{1 + \exp \langle -w, x_i \rangle} -$$

$$- \frac{1}{T} \sum_{\substack{i: y_i = w_- \\ i \in T}} \log \frac{1}{\exp \langle w, x_i \rangle + 1} = \frac{1}{T} \sum_{i=1}^T \log(1 + \exp(-y_i \langle w, x_i \rangle))$$

То есть функции потерь для бинарной логрессии и мультиномиальной регрессии при $K=2$ эквиваленты.

ч.т.д

Задание №1

Предобработка документов легко делается с помощью модуля `re` и метода `apply` из `pandas`.

```
1 X_train_df.apply(lambda x: re.sub("[^a-zA-z0-9]", " ", x.lower()))
2 X_test_df.apply(lambda x: re.sub("[^a-zA-z0-9]", " ", x.lower()))
```

Листинг 1: Clear documents

Задание №2

Использование `CountVectorizer` для представления слов с помощью `bag of words`.

```
1 vectorizer = CountVectorizer(lowercase=True, min_df=50)
2 X_train_v = vectorizer.fit_transform(X_train["comment_text"])
3 X_test_v = vectorizer.transform(X_test["comment_text"])
```

Листинг 2: Vectorizer

`Min_df = 50` имеет под собой основание. "Оскорбительные" слова в данном датасете встречаются обычно чаще 100 раз. Листинг ниже (3) демонстрирует код для подсчета.

```
1 text = X_train["comment_text"]
2 count = 0
3 count_bad = 0
4 for i in range(text.size):
5     if text[i].find("very very bad word") != -1:
6         count += 1
7         if X_train["is_toxic"][i]:
8             count_bad += 1
```

Листинг 3: Count bad words

Задание №3, №4, №5

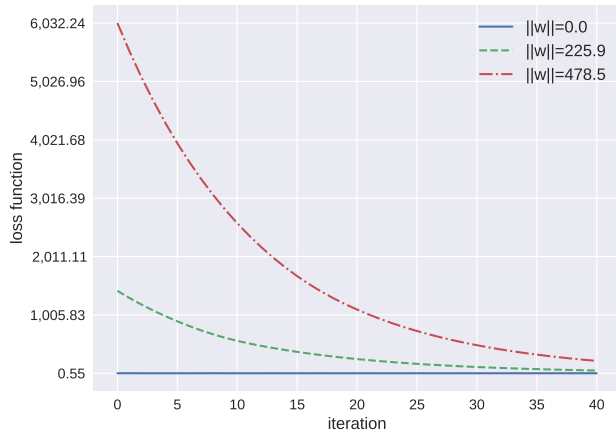
Исследуем как ведет себя метод (стохастического) градиентного спуска. Я посчитал, что 3, 4, 5 задания можно совместить в одно большое задание — легче прослеживается логика повествования.¹

Начальная инициализация

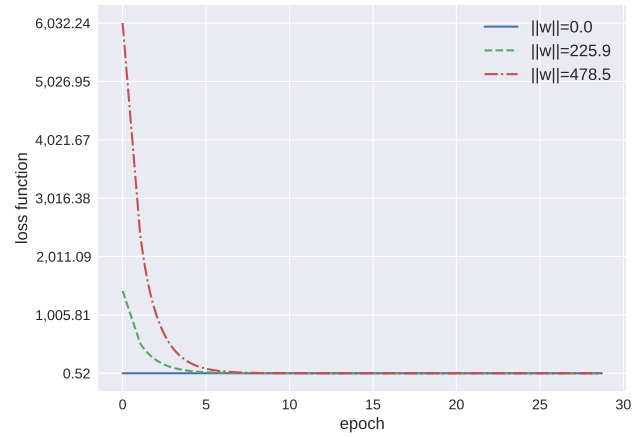
Интересно взглянуть как влияет начальная инициализация весов на функцию потерь. Таблице 1²

¹Все графики строились на весьма урезанной по количеству признаков выборке (2300). Это сделано для того, чтобы вычислительной мощности компьютера Тыцкого.В.И. хватило построить их за разумное время.

²Я так и не понял как поменять тип `caption` с Таблицы на Рисунок



GD



SGD

Таблица 1: Зависимость функции потерь от начальной инициализации и итераций

Можно заметить, что вне зависимости от начальной инициализации спустя небольшое количество итераций(эпох) функция потерь становится примерно одинаковой.

В других экспериментах будем использовать единичный вектор в качестве начальной инициализации

Параметры задающие скорость обучения

В экспериментах используется Линейных классификатор, который вычисляет новый вес w^{i+1} по формуле:

$$w^{i+1} = w^i - \eta \nabla Q, \quad \eta = \frac{\alpha}{i^\beta}$$

где Q - градиент функции потерь.

Необходимо понять как влияют гиперпараметры α и β на алгоритм. В Таблице 2 представлены графики зависимости функции потерь от параметров α и β .

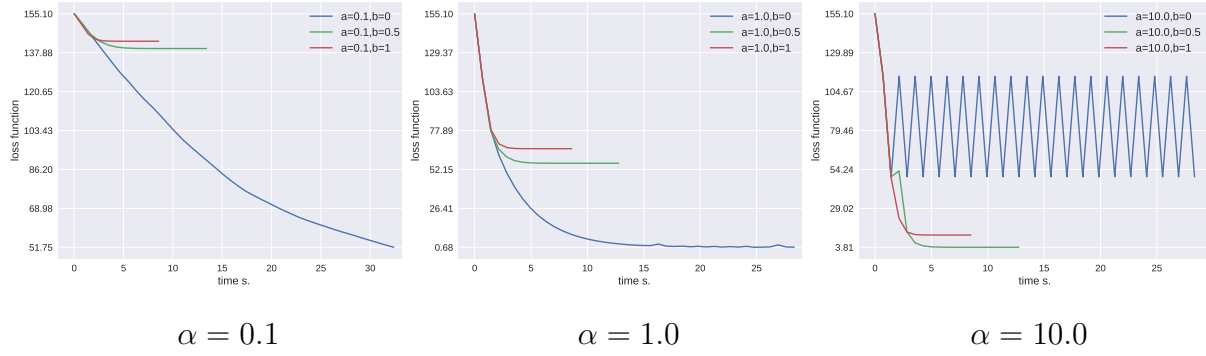


Таблица 2: Зависимость функции потерь от alpha и beta для GD

Заметим как быстро алгоритм останавливается, при $\beta > 0$. Если и имеет смысл использовать ненулевые β , то только для больших значений параметра α . В то же время при сильно больших α и $\beta = 0$ градиентный спуск может не спуститься в точку экстремума, что плохо сказывается на качестве модели.

В дальнейших экспериментах будем брать $\alpha < 1$ и $\beta = 0$.

Для стохастического градиентного спуска картина такая же (Таблица 3) за исключением того, что при больших α поведение еще более непредсказуемо.

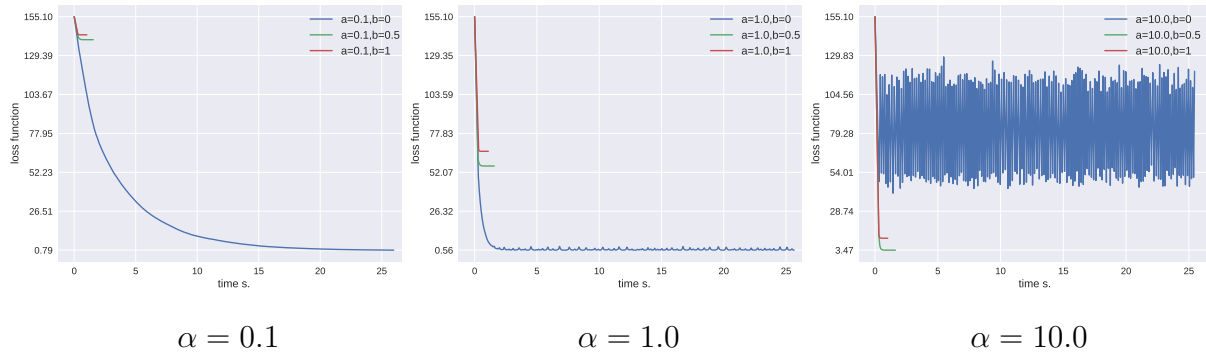


Таблица 3: Зависимость функции потерь от alpha и beta для SGD

Сравнение GD и SGD в скорости

На предыдущих графиках(Таблица 2 Таблица 3) можно пронаблюдать скорость обучения GD и SGD классификатора. GD делает более "точные" шаги градиентного спуска, но скорость выполнения этого шага довольно медленная. Хотя SGD чуть менее точен(совсем незначительно), но из-за того, что он делает больше шагов градиентного спуска за эпоху, он быстрее сходится к локальному экстремуму. Важно заметить, что выбор в пользу SGD сделан конкретно для данного датасета. Для других задач поведение GD и SGD может быть совсем разное.

В дальнейших экспериментах будем использовать SGD классификатор.

Время на одну итерацию(эпоху)

Важно оценить время работы классификатора в зависимости от итераций(эпох), чтобы подобрать оптимальное по соотношению качество скорость итераций(эпох). Таблица 4. Одна итерация(эпоха) делается чуть меньше, чем за секунду.

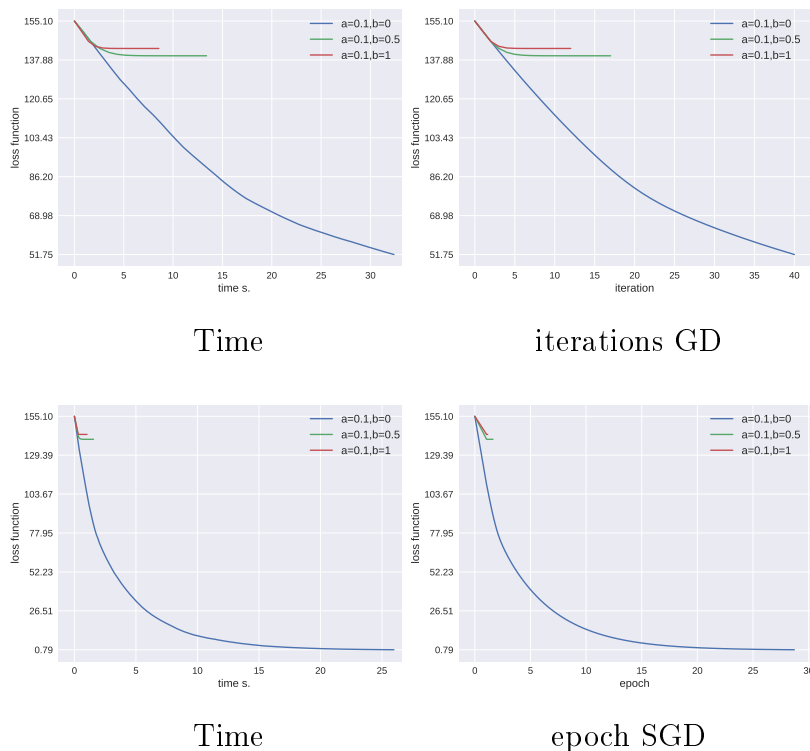


Таблица 4: Зависимость между временем и итерацией(эпохой)

Размер батча для SGD

В случае выбора SGD в качестве основного алгоритма важно понять какой размер батча(подвыборки) оптимален. Важна и скорость работы, и точность шагов градиентного спуска. Справа представлен график Рис.1.

Можно заметить, что уже при размерах батча 500, 2000 скорость сходимости и точность шагов градиентного спуска приемлемы — не возникает скачков, как у размера 100 и скорость гораздо выше, чем для размера 10000 или всей выборки.

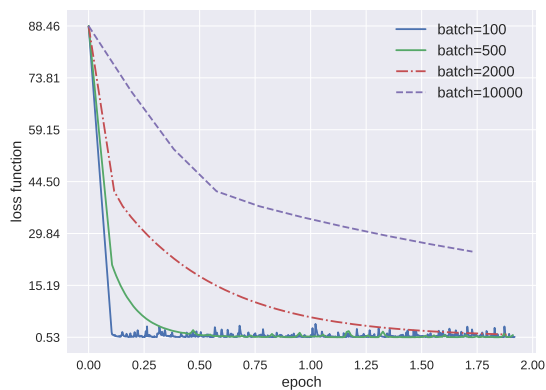


Рис. 1: Зависимость функции потерь от размера батча

В угоду точности сходимости можно немного пожертвовать скоростью, поэтому в будущих экспериментах будет использоваться размер батча 2000.

Качество алгоритма

Уменьшение функции потерь — не самое важное для нас. Необходимо взглянуть как меняется мера качества в идеале на отложенной выборке.³ Ниже представлены графики Таблица 5 для SGD классификатора ($\text{batch_size}=2000$, $\text{l2_coef}=0.1$)

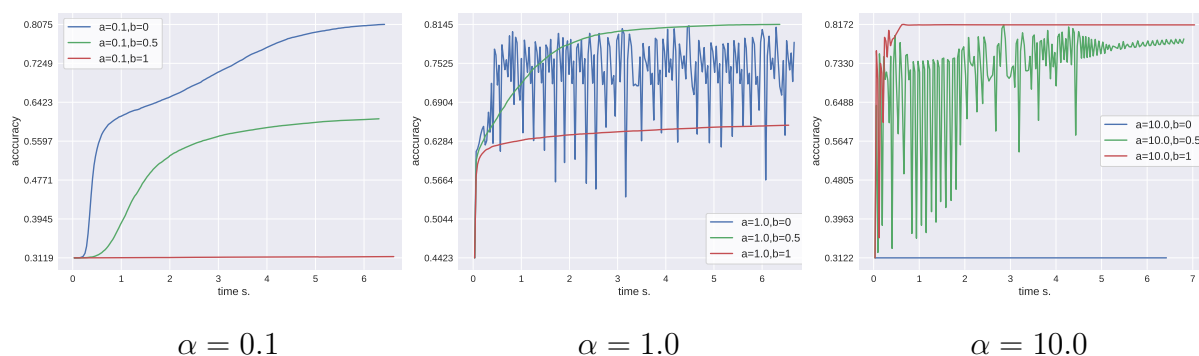


Таблица 5: Зависимость функции потерь от alpha и beta для GD

Лучшее качество на данном этапе экспериментов для тестовой выборки: **roc-auc— 0.928, accuracy— 0.860**

Обучение проводилось на выборке, векторизованной Bag of words с $\text{min_df} = 50$, с помощью SGD классификатора с параметрами:

$\text{batch_size} = 2000, \alpha = 0.5, \beta = 0, \text{max_iter} = 1500, \text{l2_coef} = 0.01, \text{tolerance} = 1e-7, \text{fit_intercept} = \text{True}.$

Задание №6

При обработке текстов возникает проблема того, что одно и то же слово может присутствовать в коллекции в разных формах (например play, played). Все слова стоит привести в начальную форму — это может существенно уменьшить признаковое пространство и улучшить качество модели. Возьмем за основу алгоритм лемматизации слов. Ниже представлен Листинг 4, демонстрирующий применение алгоритма лемматизации с помощью библиотеки nltk.

```
1 # https://webdevblog.ru/podhody-lemmatizacii-s-primerami-v-python/
2 def get_wordnet_pos(word):
3     tag = nltk.pos_tag([word])[0][1][0].upper()
4     tag_dict = {"J": wordnet.ADJ,
5                "N": wordnet.NOUN,
6                "V": wordnet.VERB,
```

³Не стал делать отложенную выборку и замерил на обучающей

```

7         "R": wordnet.ADV}
8     return tag_dict.get(tag, wordnet.NOUN)
9
10
11 lem= WordNetLemmatizer()
12
13 X_train.apply(lambda x: " ".join([lem.lemmatize(w, get_wordnet_pos(w))
14     for w in nltk.word_tokenize(x)]))
15 X_test.apply(lambda x: " ".join([lemm.lemmatize(w, get_wordnet_pos(w))
16     for w in nltk.word_tokenize(x)]))

```

Листинг 4: Lemmatizer

Также перед лемматизацией удалим стоп-слова.

Размерность признакового пространства снизилась с 3930 до 3090. На скорость работы уменьшение размерности не сильно повлияло⁴. Accuracy и roc-auc на тестовой выборке существенно возросло: **roc-auc – 0.935, accuracy – 0.865**

Обучение проводилось на лемматизированной выборке, векторизованной Bag of words с $min_df = 50$, $stop_words = english$, с помощью SGD классификатора с параметрами:

$batch_size = 2000$, $\alpha = 0.8$, $\beta = 0$, $max_iter = 1800$, $l2_coef = 0.01$, $tolerance = 1e - 7$, $fit_intercept = False$.

⁴в виду того, что и так размерность не самая большая

Задание №7

Посмотрим как влияет предобработка данных, а конкретно параметры `min_df`, `max_df`. В Таблице 6 представлены зависимости времени, размерности, качества(BoW, TFIDF) от `min_df`, `max_df`.

		shape	time s.	accuracy BoW	accuracy TFIDF
<code>max_df = max</code>	<code>min_df = 10</code>	11000	23	0.851	0.741
	<code>min_df = 100</code>	2080	7.5	0.850	0.831
	<code>min_df = 1000</code>	150	2.0	0.784	0.784
<code>min_df = 50</code>	<code>max_df = 300</code>	2957	8.27	0.723	0.698
	<code>max_df = 800</code>	3452	9.59	0.801	0.725
	<code>max_df = max</code>	3660	10.79	0.848	0.766

Таблица 6: Зависимость некоторых величин от `min_df`, `max_df` и типов векторизации

Размерность пространства очень сильно увеличивается, если `min_df` близко к нулю, но качество в свою очередь несильно меняется. Как было упомянуто в начале исследования, обычно все 'плохие' слова встречаются чаще, чем 100 раз на коллекцию, поэтому уменьшение `min_df` приводит только к увеличению размерности и времени работы.

`Max_df` неразумно ставить каким-либо, кроме максимального в данном случае т.к. мы можем потерять важную информацию, а самые распространенные слова(стоп-слова) английского языка мы предварительно убрали.

Интересно, что качество после обработки с помощью TFIDF стабильно хуже, чем BoW. **Лучший алгоритм показал roc-auc – 0.935, accuracy – 0.853 на тестовой выборке.(TF-IDF)**