Исследование работы линейных моделей

Тыцкий Владислав

Ноябрь 2020

Градиент в логистической регрессии

Пусть $X \in R^{NxF}$ – 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 константый признак.

Дана задача оптимизации:

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

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

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

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 $X \in R^{NxF}$ – 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:

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

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

Найдем градиент по w_m.

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

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

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

Отсюда получаем:

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

Эквивалетность 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 логрегрессии 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.

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

```
vectorizer = CountVectorizer(lowercase=True, min_df=50)
X_train_v = vectorizer.fit_transform(X_train["comment_text"])
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.

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

Листинг 3: Count bad words

Задание №3, №4, №5

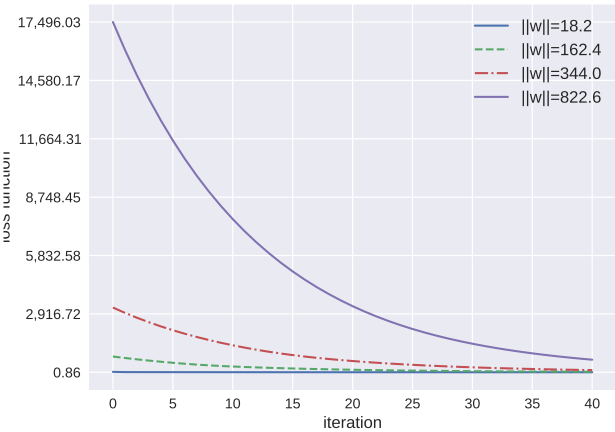
Исследуем как ведет себя метод (стохастического) градиентого спуска. Я посчитал, что 3, 4, 5 задания можно совместить в одно большое задание — легче прослеживается логика повествования. [1]

Начальная инициализация

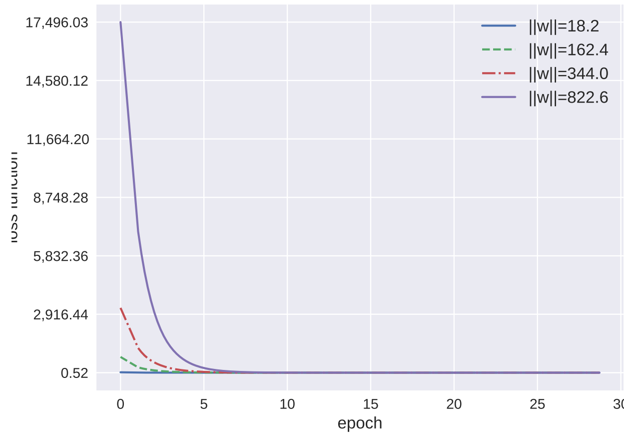
Интересно взглянуть как влияет начальная инициализация весов на функцию потерь.Таблице 1 [2]

[1] Все 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.

[2] Я 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 внезависисмости от начальной иниципализации спустя небольшое количество итераций(эпох) функция потерь становится примерно одинаковой.

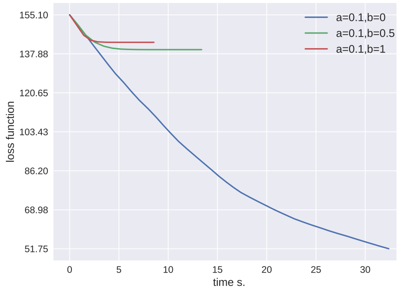
В других экспериментах будем использовать единичный вектор в качестве начальной инициализации

Параметры задающие скорость обучения

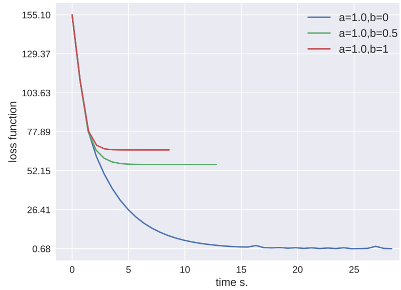
В эсперherметнах используется Линейных классификатор, который вычисляет новый вес w^{i+1} по формуле:

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

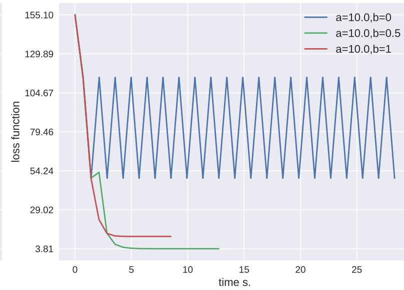
где Q - градиент функции потерь.
Небоходимо понять как влияют гиперпараметры α и β на алгоритм. В Таблице 2 представлены графики зависимости функции потерь от параметров α и β.



$\alpha = 0.1$



$\alpha = 1.0$



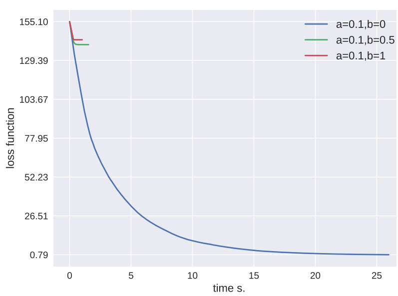
$\alpha = 10.0$

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

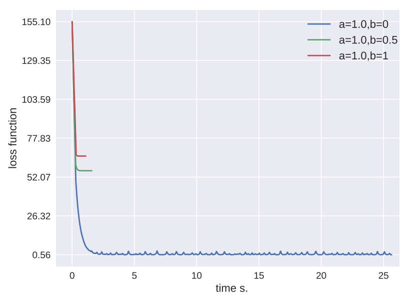
Заметим как быстро алгоритм отстанавливается, при $\beta > 0$. Если и имеет смысл использовать ненулевые β, то только для больших значений параметра α. В то же время при сильно больших α и $\beta = 0$ градиентный спуск может не спуститься в точку экстремума, что плохо сказывается на качестве модели.

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

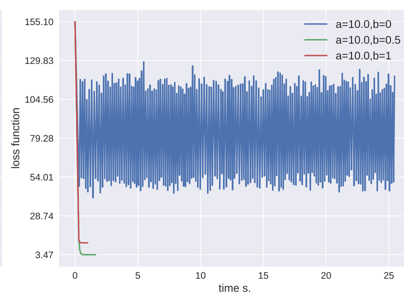
Для стохастического градиентного спуска картина такая же (Таблица 3) за исключением того, что при больших α поведение еще более непредсказуемо.



$\alpha = 0.1$



$\alpha = 1.0$



$\alpha = 10.0$

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

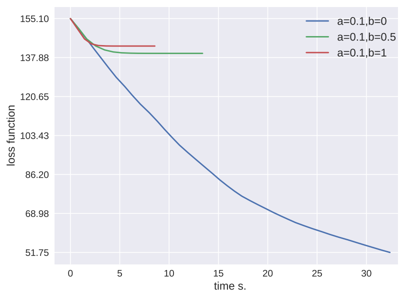
Сравнение GD и SGD в скорости

На предущих 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. Хоть 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.

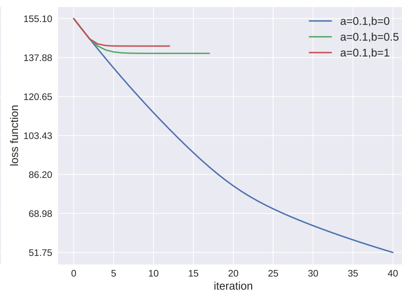
В дальнейших экспериментах будем использовать SGD классификатор.

Время на одну итерацию(эпоху)

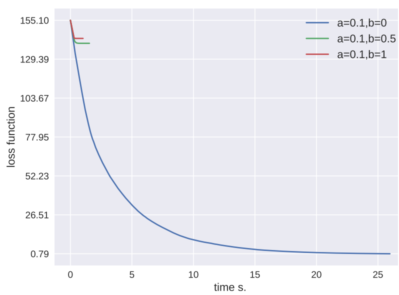
Важно оценить время работы классификатора в зависимости от итераций(эпох), чтобы подобрать оптимальное по соотношению качество скорость итераций(эпох). Таблица 4. Одна итерация(эпоха) делается чуть меньше, чем за секунду.



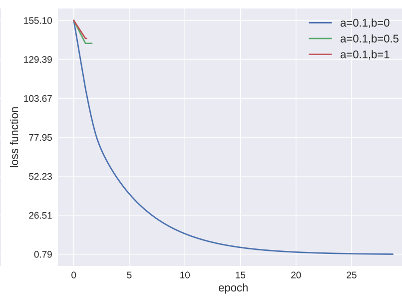
Time



iterations GD



Time



epoch SGD

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 градиентого спуска приемлимы — не возникает скачков, как у размера 100 и скорость гораздо выше, чем для размера 10000 или всей выборки.

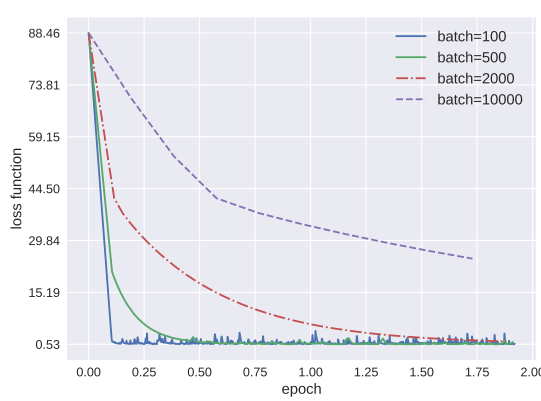
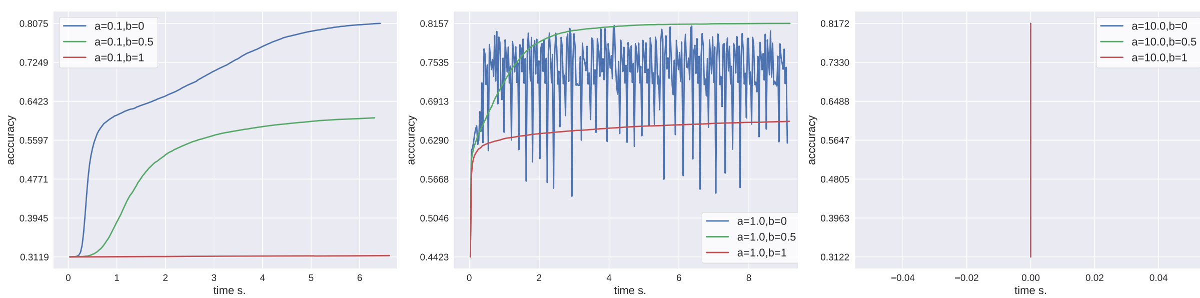


Рис. 1: Зависимость функции потерь от размера батча

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

Качество на обучающей выборке

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, batch_size=2000, l2_coef=0.1



$\alpha = 0.1$ $\alpha = 1.0$ $\alpha = 10.0$

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

[3] Не стал делать отложенную выборку и мерил на обучающей