В втором практическом задании требуется реализовать методы решения задачи логистической регрессии для решения задачи бинарной классификации. Дано N пар (x_i, y_i), где $x \in \mathbb{R}^n$, $y_i \in [0, 1]$. x принятно называть наблюдениями (примерами, etc), y_i лейблами/метками/etc. В случаи логистической регрессии предполагается, что y_i это вероятность того, что наблюдение x_i относится к первому классу

Рассмативается следующая модель зависимости y от x:

$$p(y = 1|x) = \sigma_w(x) = \frac{e^{f_w(x))}}{1 + e^{f_w(x)}}, \tag{1}$$

где

$$f_w(x) = w^T x \tag{2}$$

В первом практическом задании требуется реализовать методы решения задачи логистической регрессии для решения задачи бинарной классификации. Дано N пар (x_i, y_i), где $x \in \mathbb{R}^n$, $y_i \in [0, 1]$. x принятно называть наблюдениями (примерами, etc), y_i лейблами/метками/etc. В случаи логистической регрессии предполагается, что y_i это вероятность того, что наблюдение x_i относится к первому классу

Рассмативается следующая модель зависимости y от x:

$$p(y = 1|x) = \sigma_w(x) = \frac{e^{f_w(x))}}{1 + e^{f_w(x)}}, \tag{3}$$

где

$$f_w(x) = w^T x \tag{4}$$

Для нахождения параметра w предлагается решать следующую оптимизационную задачу:

$$w = \operatorname*{arg\,max}_{w \in \mathbb{R}^n} \frac{1}{N} \sum_{i=1}^{N} y_i log(\sigma_w(x_i)) + (1 - y_i) log(1 - \sigma_w(x_i)) \tag{5}$$

Таким образом, ваша задача — найти минимум функции

$$F_{train}(w) = -\frac{1}{N} \sum_{i=1}^{N} y_i log(\sigma_w(x_i)) + (1 - y_i) log(1 - \sigma_w(x_i)) \tag{6}$$

по параметру w.

Замечание: в описании модели я предполагаю, что в x последняя координата тождественна равна 1. Первые $n-1$ компонент вектора это признаки, с помощью которых описывается зависимость, последняя для удобного задания константной поправки. Для тех, кто не знаком с таким способом записи — представьте обычную линейную зависимость $\alpha x_0 + \beta$. Здесь 2 параметра, их надо искать. Для записи в векторном виде удобно добавить dummy-переменную x_1, всегда равную 1

Решения необходимо реализовать на Python 3.6+, используя scipy и numpy.

Исследования рекомендуется делать в jupyter-notebook'ах с экспортом в читаемый PDF (Необходимо, что бы я мог прочитать все графики и комментарии к ним + при необходимости запустить код ячеек)

Основное требование от кода — он должен работать без установки дополнительных пакетов, кроме тех, которые поставляются в anaconda + можно использовать plotly для построения графиков.

Пункт 1

Записать задачу в матричном виде, выписать формулы для вычисления значения функции в векторном виде, а также для для вычисления значения градиента и гессиана этой функции.

Ответить на следующие вопросы: будет ли зависеть результат работы метода для логистической регрессии от выбора начальной точки? если да, то почему, если нет, то почему, каким образом будет зависить?

Пункт 2 Реализовать оракулы для задачи логистической регрессии: оракул нулевого порядка (вычисляет функцию), оракул первого порядка (функцию + градиент) и оракул второго порядка (функция + градиент + гессиан).

Для реализованных оракулов требуется написать тест, проверяющий корректность вычисления вычисленных производных с помощью разностного дифференцирования

```
#format = "libsvm" or "tsv", tsv —нулевойстолбецлейбл , остальноепризнаки
def make_oracle(data_path, format="libsvm"):
        return oracle

#oracle looks like this
def f(x, need_grad=True, need_hessian=True):
        result = (f(x))
        if need_grad:
                result.append(grad(f, x))
    return result
```

Пункт 3

Реализовать метод градиентного спуска для произвольного метода поиска по прямой

```
def line_search(f, x, direction):
        return lambda

def optimize(f, start_point, line_search_method, tol=1e-8, max_iter=10000)
     x = ...
        return x
```

В качестве критерия остановки использовать

$$\frac{|\nabla F(w_k)|^2}{|\nabla F(w_0)|^2} \leq \varepsilon \tag{7}$$

Изучить эмпирически скорость работы на нескольких наборах данных (наборы данных смотрите после описание всех заданий) следующих стратегий линейного поиска в методе градиентного спуска:

1. наискорейший спуск, в качестве метода одномерной реализации использовать реализованный вами метод золотого сечения
2. наискорейший спуск с использованием метода Брэнта
3. Неточный поиск шага с использованием условия армихо. Для этого необходимо реализовать поиск шага с помощью условий Армихо. Также необходимо поэксперементировать с разными параметрами для этого метода (и написать выводы по результатам + результаты 2-3 запусков)
4. Неточный поиск с условиями Вульфа, используйте реализацию из scipy. Точно так же изучите влияние параметров и напишите выводы + результаты 2-3 запусков
5. Выбор шага с помошью алгоритма, оценивающего локальную константу Липшеца (было на паре)

Для каждой стратегии необходимо построить графики зависимости $log(r_k)$ от

- Времени работы метода
- Числа вызовов оракула
- Числа итераций метода

где r_k одна из следующих последовательностей:

- $r_k = |F(w_k) - F_(w_*)|$, где w_* оптимальное решение, в которое сходится метод из точки w_0. Для нахождения оптимального решения можно воспользоваться реализованными в numpy/scipy методами оптимизации (чтобы не сравниваться с ошибками, которые возникнут у вас в коде) от числа итераций
- $r_k = \frac{|\nabla F(w_k)|^2}{|\nabla F(w_0)|^2}$

Пункт 5 Реализовать метод Ньютона. Провести аналогичные прошлому пункту (т.е. все) исследования по стратегиям выбора шага для реализованного метода, кроме оценки константы Липшица

Пункт 6 Реализовать Hessian-free Newton для решения логистической регресии, т.е. метод ньютона, в котором решение системы $H_k d = g_k$ производится с помощью метода сопряженных градиентов + неточно, при этом точность решения увеличивается в процессе оптимизации + в сопряженных градиентах матрица H_k явно не строится, а только делается умножение гессиана на вектор.

```
def hfn_optimize(f, init_point, tolerance, policy="sqrtGradNorm", tolerance_eta=
    return optimal_point
```

Исследовать работы Hessian-Free Newton

1. изучить эмпирически скорость работы на нескольких наборах данных в зависимосьти от стратегий выбора точности, с которой решается задача в методе CG

2. Сравнить с методом Ньютона и градиентным спуском ("оптимальными" версиями, полученными при выполнении предыдущих пунктов)

Внимание: ваш код должен работать с сильно разреженными данными. В качестве проверки работы метода ваш скрипт будет запущен на сильно разреженном датасете, которые не будет помещаться в виде dense матрицы в память ноутбука

Данные находятся в libsvm-формате, их можно загружать в питон с помощью sklearn.datasets.load_svmlight_file

Внимание: метки y в данных датасетах скорее всего равны $-1, 1$. Вам необходимо будет их переделать в $0, 1$, как определено в нашей модели

Оценка за задания

Я допишу чуть позже формальные критерий, базова оценка за весь курс будет складываться из оценок по методам (градиентный спуск + метод ньютона, NFN, будущие) Для 3 достаточно, что бы код работал на моих данных с заданной точностью (хотя бы с одной фиксированной стратегией линейного поиска) Для 4 необходимо наличие подробного исследования разницы работы методов с разными параметрами (line search, сравнение с другими методами, etc) и комментариев к тому, что смотрели и зачем, какие выводы сделали Для 5 необходимо будет на экзамене по моим вопросам к реализованым методам уметь объяснять, почему сделано так или иначе, понимать что будет, если поменяют такой то или такой то вход, внесу такие-то изменения в оракул и т.п.

Необходимо
, чтобыпримернотакойкодвконце ipython сзаданиемработал
:

```
\begin{lstlisting}[language=Python]

oracle = make_oracle("some_path.libsvm")
best_point = hfn_newton(oracle, my_start_point, tol=my_tolerance)
assert abs(my_loss_implementation(best_point) - optimal_loss) < my_tolerance
```

Для тестирования необходимо использовать наборы данных из libsvm-репозитория https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html: и один симмулированный пример

- Adult (a1a)

- Breast cancer (breast-cancer)

- Сгенерируйте случайный датасет из 1000 наблюдений для зависимости $\alpha{*}x{+}\beta$, где x случайная нормальная величина с средним 0 и дисперсией 1, $y = 1$ если $a * x + b >= 0$ и ноль иначе. Параметры α, β выберете случайно из отрезка $[-1, 1]$

Данные находятся в libsvm-формате, их можно загружать в питон с помощью sklearn.datasets.load_svmlight_file

Дедлайн сдачи задания: 18 ноября в 8 вечера Дедлайн сдачи задания: до 22 ноября в 8 вечера максимальная оценка умножается на 0.8 Дедлайн сдачи задания: до 28 ноября в 8 вечера максимальная оценка умножается на 0.6 Дедлайн сдачи задания: после 28 ноября, 8 вечера максимальная оценка умножается на 0.