**Федеральное агентство по образованию**Государственное образовательное учреждение высшего профессионального   
образования **«Тихоокеанский Государственный университет»**

Факультет компьютерных и фундаментальных наук

Кафедра ПОВТАС

**Лабораторная работа №2**

по дисциплине: «Методы машинного обучения»

на тему: «Обучение линейного алгоритма бинарной классификации образов с

помощью градиентного алгоритма»  
Вариант №4

Выполнил: студент группы ПИИ(м)-21

Латынцев А.В.

Проверил: преподаватель кафедры ПОВТАС

Тормозов В.С.

# Постановка задачи

**Цель работы**: научиться реализовывать алгоритм градиентного спуска для задачи обучения линейной модели бинарной классификации образов.

**Задания на лабораторную работу** (8 вариант)

В файле iris\_data.py даны обучающие выборки (по вариантам) для обучения линейного алгоритма бинарной классификации образов:

http://tk.ulstu.ru/files/iris data.py

Модель линейного алгоритма должна иметь вид:

,

где - весовые коэффициенты модели (определяют ориентацию разделяющей линии); - образ обучающей выборки;

– знаковая функция

То есть, метки классов принимают значения

Ваша задача **выполнить обучение линейной модели** (найти значения вектора весовых коэффициентов ) с помощью градиентного алгоритма (программы, написанной на языке Python), который должен минимизировать величину эмпирического риска:

где - нотация Айверсона (квадратные скобки возвращают 1, если условие в скобках истинно, и 0 - в противном случае). То есть, эмпирический риск показывает число неверных классификаций.

Так как градиентный алгоритм может минимизировать только гладкие, дифференцируемые функции, то величину следует сверху ограничить именно таким функционалом:

где - выбранная функция потерь (здесь – отступ).

Функция потерь (также, как и набор обучающих данных) определяется вариантом. Для 8 варианта:

|  |  |
| --- | --- |
| Функция потерь для реализации градиентного алгоритма | Производная функции потерь |
| экспоненциальная |  |

В качестве начальных значений весовых коэффициентов можно взять следующие:

Шаг в градиентном алгоритме для коэффициента целесообразно выбрать побольше, а для коэффициентов - поменьше.

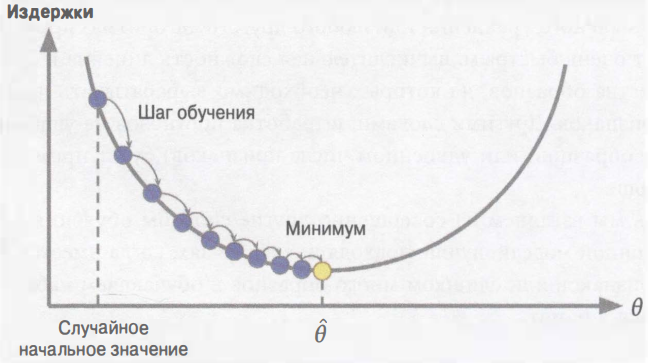
**Построить график** множества точек обучающей выборки (каждый класс точек должен быть представлен разными маркерами и цветами) и полученной разделяющей линии.

# Краткая теория

Градиентный спуск представляет собой самый общий алгоритм оптимизации, способный находить оптимальные решения широкого диапазона задач. Основная идея градиентного спуска заключается в том, чтобы итеративно подстраивать параметры для сведения к минимуму функции издержек.

Градиентный спуск выражается в измерении локального градиента функции ошибок применительно к вектору параметров и движении в направлении убывающего градиента. Как только градиент становится нулевым, был достигнут минимум!

Выражаясь более конкретно, все начинается с наполнения вектора случайными значениями (т.е. случайная инициализация). Затем происходит постепенное его уточнение по одному маленькому шагу за раз и на каждом шаге снижается функция издержек (например, MSE) до тех пор, пока алгоритм не сойдется в минимуме:



Важным параметром в градиентном спуске является размер шагов, определяемый гиперпараметром скорости обучения. Если скорость обучения слишком мала, тогда алгоритму для сведения придется пройти через множество итераций, что потребует длительного времени:

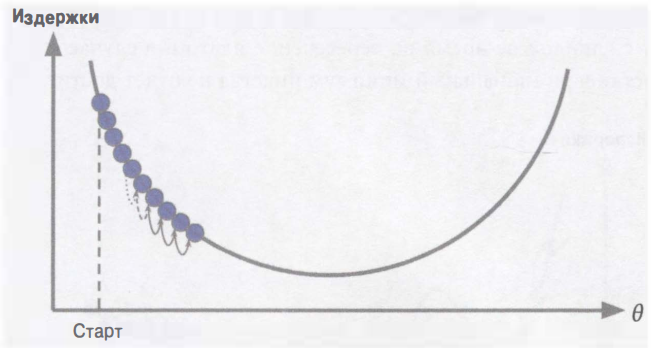


Рис. . Скорость обучения слишком мала

С другой стороны, если скорость обучения слишком высока, тогда можно перескочить «долину» и оказаться на другой стороне, возможно даже выше, чем ранее. Это способно сделать алгоритм расходящимся, что приведет к выдаче постоянно увеличивающихся значений и неудаче в поиске хорошего решения:

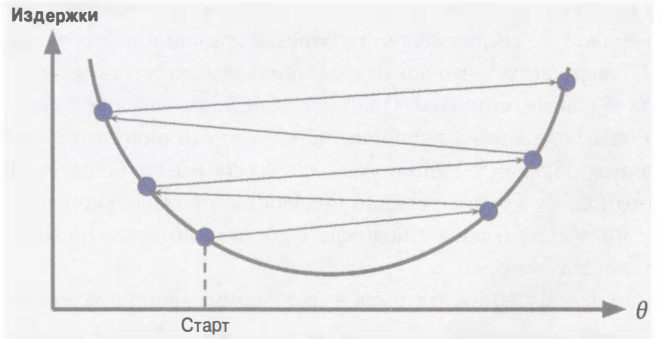
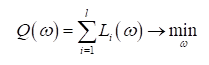
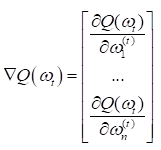


Рис. . Скорость обучения слишком высока

Как известно, эмпирический риск:



В этом случае мы имеем многомерную функцию и на каждой итерации нам нужно вычислять частные производные по каждому из значений:



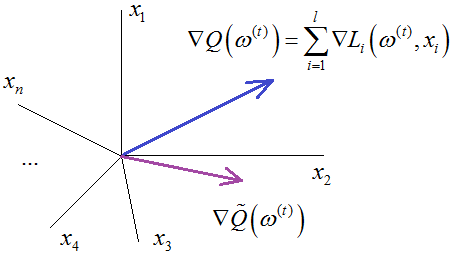
Получим вектор из частных производных, которым, затем, уточним значение вектора коэффициентов на новой итерации:



Если расписать эту формулу еще подробнее, то вместо градиента функционала у нас получится сумма градиентов функции потерь по обучающей выборке:



Получается, чтобы сделать один шаг классического градиентного спуска, нужно вычислить производные по функциям потерь для всех объектов обучающей выборки и сложить их, что ресурсоемко. Поэтому при практической реализации исходный градиент (всю сумму целиком) заменяют псевдоградиентом (субградиентом), который вычисляется значительно проще. И единственное требование к псевдоградиенту – его направление должно в среднем, на каждом шаге, образовывать острый угол с истинным градиентом.



Здесь через  обозначен псевдоградиент.

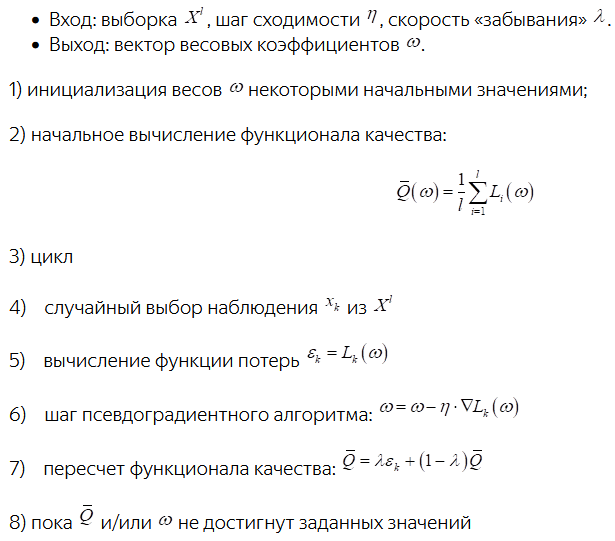
Возникает вопрос, что нам взять в качестве псевдоградиента, то есть, как сократить объем вычислений для истинного градиента и при этом сохранить сходимость алгоритма к точке локального минимума? Самое простое и радикальное, что можно сделать, на каждом шаге из всей суммы брать только одно наблюдение (одно слагаемое) и по нему вычислять текущее приближение:



где  - псевдоградиент; k – случайный индекс вектора из обучающей выборки.

Будет ли в этом случае наш псевдоградиент образовывать в среднем острый угол с истинным градиентом? Другими словами, будем ли мы в среднем двигаться в направлении локального минимума функционала качества? Да, можно показать, что по закону больших чисел, при большом числе шагов (приближений) этот псевдоградиент сходится к истинному градиенту. А, значит, рано или поздно мы достигнем с его помощью точки минимума.

Эта идея положена в основу метода стохастического градиентного спуска (Stochastic Gradient Descent – SGD). Иногда его еще называют методом Роббинса-Монро. Алгоритм можно записать в виде следующего псевдокода:



# Результаты работы

Работа была выполнена на языке программирования Python 3 с использованием Jupyter Notebook.

Первым делом (Листинг 1):

|  |
| --- |
|  |

Листинг 1.

Следующий шаг –

|  |
| --- |
|  |

Листинг 2.

Листинг 3 описывает

|  |
| --- |
|  |

Листинг .

В Листинг 4 описан

|  |
| --- |
|  |

Листинг .

Класс

|  |
| --- |
|  |

Листинг .

Класс

|  |
| --- |
|  |

Листинг 6.

Наконец в Листинг 7 приведен код

|  |
| --- |
|  |

Листинг .

В

# Вывод

В ходе лабораторной работы я выполнил обучение линейной модели (нашел значения вектора весовых коэффициентов ) с помощью градиентного алгоритма, построил соответствующие графики.