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

Факультет компьютерных и фундаментальных наук

Кафедра ПОВТАС

**Лабораторная работа №3**

по дисциплине: «Методы машинного обучения»

на тему: «Исследование работы L2-регуляризатора в задачах регрессии»  
Вариант №4

Выполнил: студент группы ПИИ(м)-21

Забавин А.С.

Проверил: преподаватель кафедры ПОВТАС

Тормозов В.С.

# Постановка задачи

**Цель работы**: изучить особенности работы L2-регуляризатора на примере задачи аппроксимации функции линейной моделью.

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

1. Вам необходимо аппроксимировать (описать) функцию своего варианта с помощью линейной модели:

то есть, полиномом 13-й степени. Здесь - весовые коэффициенты, которые требуется найти с помощью градиентного алгоритма по обучающему набору данных.

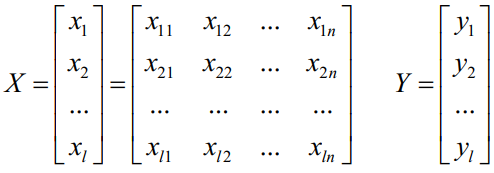
1. Обучающую выборку следует составить из всех четных индексов сгенерированных значений функции:

То есть, сначала формируется первое значение с целевым значением , затем, второе: и так пока не дойдем до конца диапазона.

1. После этого, вычислите значения коэффициентов вектора для квадратической функции потерь (в задачах регрессии, обычно, используют именно такую функцию потерь), которые минимизируют эмпирический риск:

Коэффициенты вычисляются по формуле:

где - входные векторы обучающей выборки; - вектор (или матрица) целевых значений обучающей выборки:



1. Вычислите прогнозы функции с помощью полученной модели для всего диапазона значений. (В отсчетах, не участвующих в выборке, значения модели должны сильно расходиться с целевыми.)
2. Вычислите коэффициенты вектора с L2 регуляризатором по формуле:

где - коэффициент регуляризации; - единичная матрица.

1. Для новой модели повторите вычисление прогнозов функции для всего диапазона значений.

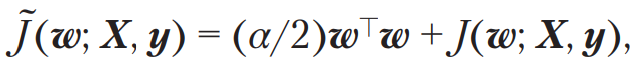
Все программы реализовать на языке Python с использованием пакетов NumPy и Matplotlib.

Функция для исследования L1 и L2 регуляризаторов (8 вариант):

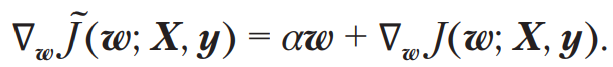
# Краткая теория

Один из самых простых и распространенных видов штрафа параметров – по норме L2, – часто называют снижением весов. Цель такой стратегии регуляризации – выбирать веса, близкие к началу координат, за счет прибавления к целевой функции члена регуляризации . Регуляризацию по норме L2 называют также **гребневой регрессией**, или **регуляризацией Тихонова**.

Получить качественное представление о поведении регуляризации методом снижения весов можно, изучив градиент регуляризированной целевой функции. Для простоты опустим параметр смещения, т. е. будем считать, что совпадает с . Полная целевая функция в такой модели имеет вид:



а градиент по параметрам:



Один шаг обновления весов с целью уменьшения градиента имеет вид:



То же самое можно переписать в виде:



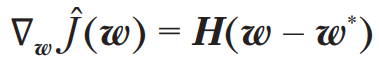
Как видим, добавление члена сложения весов изменило правило обучения: теперь мы на каждом шаге умножаем вектор весов на постоянный коэффициент, меньший 1, перед тем как выполнить стандартное обновление градиента. Итак, мы описали, что происходит на одном шаге. А в целом в процессе обучения?

Еще упростим анализ, предположив квадратичную аппроксимацию целевой функции в окрестности того значения весов, при котором достигается минимальная стоимость обучения без регуляризации, . Если целевая функция действительно квадратичная, как в случае модели линейной регрессии со среднеквадратической ошибкой, то такая аппроксимация идеальна. Аппроксимация описывается формулой:



где – матрица Гессе относительно , вычисленная в точке . В этой квадратичной аппроксимации нет члена первого порядка, потому что по определению, точка минимума, в которой градиент обращается в нуль. Из того, что – точка минимума , следует также, что матрица положительно полуопределенная.

Минимум достигается там, где градиент



равен 0.

До сих пор мы обсуждали снижение весов в терминах воздействия на оптимизацию абстрактной квадратичной функции стоимости. Но как эти эффекты проявляются конкретно в машинном обучении? Это можно выяснить на примере изучения линейной регрессии – модели, в которой истинная функция стоимости квадратичная и потому поддается проведенному выше анализу. Повторяя те же рассуждения, мы получим для этого частного случая результат, сформулированный в терминах обучающих данных. Для линейной регрессии функция стоимости равна сумме квадратов ошибок:



После добавления L2-регуляризации целевая функция принимает вид:



В результате нормальные уравнения, из которых ищется решение:



принимают вид



Матрица в уравнении пропорциональна ковариационной матрице . Применение L2-регуляризации заменяет эту матрицу на в последнем уравнении. Новая матрица отличается от исходной только прибавлением ко всем диагональным элементам. Диагональные элементы этой матрицы соответствуют дисперсии каждого входного признака. Таким образом, L2-регуляризация заставляет алгоритм обучения «воспринимать» вход как имеющий более высокую дисперсию и, следовательно, уменьшать веса тех признаков, для которых ковариация с выходными метками мала, по сравнению с добавленной дисперсией.

# Результаты работы

Работа была выполнена на языке программирования Python 3. Код программы представлена на Листинге 1.

import numpy as np

import matplotlib.pyplot as plt

#===============================================================================

# Функция для исследования

#===============================================================================

# вариант 5

# x принадлежит множеству [0;10; 0,1]

x = np.arange(0, 10.1, 0.1) # 10.1l для захватывания крайей точки

def **y\_from\_x**(x): # функция в виде полинома -0.1x^5 + 5x^4 - 700x^2

return (-0.1 \* x\*\*5 + 5 \* x\*\*4 - 700 \* x\*\*2)

y = np.array([y\_from\_x(x\_) for x\_ in x])

#===============================================================================

#===============================================================================

# Обучающая выборка

#===============================================================================

x\_train, y\_train = x[::2], y[::2] # Четные точки - это обучающая

#===============================================================================

# Необходимо провести исследование полиноминальной функции

# Апроксимируем функцию полниномом Np степени

# Значения степени полинома ========================

Np = 12 # по заданию

# ==================================================

# numpy подбирает для обучающей выборки подходящие коэффициенты полиномов

z\_train = np.polyfit(x\_train, y\_train, Np)

#===============================================================================

# numpy имеет встроенный генератор функции предикта по минимизации вектора ошибки

# она под капотом выглядит примерно так:

# def predict\_poly(x, koeff):

# res = 0

# xx = [x \*\* (len(koeff) - n - 1) for n in range(len(koeff))]

# for i, k in enumerate(koeff):

# res += k \* xx[i]

# return res

#===============================================================================

predict\_poly\_numpy = np.poly1d(z\_train)

poly\_koefs = z\_train

y\_func\_repr = [np.round(yy, 5) for yy in y]

y\_func\_repr = y\_func\_repr[:3] + [*'...'*] + y\_func\_repr[-3:]

predict\_numpy\_repr = [np.round(np.poly1d(z\_train)(xx), 5) for xx in x]

predict\_numpy\_repr = predict\_numpy\_repr[:3] + [*'...'*] + predict\_numpy\_repr[-3:]

print(*f'Коэффициенты полинома: \n{poly\_koefs}'*)

print(*f'Исходные: y= {y\_func\_repr}'*)

print(*f'Предсказание по {Np}-полиноминальной модели: y= {predict\_numpy\_repr}'*)

# размер признакового пространства (степень полинома Np+1) (+ w0)

N = Np + 1 # размер признакового пространства (степень полинома N-1)

# Т.к. мы используем L2 регуляризацию то производная функционала качества (по вектору w) будет иметь вид w\* = (XT·X + lm·I)^-1 · XT·Y

# Поэтому в задаче стоит подбирать lm коэффициент, это и есть L2 регуляризация

# Значения коэфициента регуляризации ===============

lm = 21 # при увеличении N увеличивается lm (кратно): 12; 0.2 13; 20 15; 5000

# ==================================================

X = np.array([[a \*\* n for n in range(N)] for a in x]) # матрица входных векторов

lmI = np.array([[lm if i == j else 0 for j in range(N)] for i in range(N)]) # матрица lm\*I, где I - единичная матрица

lmI[0][0] = 0 # первый коэффициент не регуляризуем (w0)

X\_train = X[::2] # обучающая выборка

Y = y\_train # обучающая выборка

# вычисление коэффициентов по формуле w\* = (XT·X + lm·I)^-1 · XT·Y

A = np.linalg.inv(X\_train.T @ X\_train + lmI) # Инвариантная матрица

w = Y @ X\_train @ A # @ - 3-питоновский синтаксический сахар для операции numpy.matmul(), перемножение матриц

print(*f'Весовые коээфициенты (отрегуляризированные): \n{w}'*)

# отображение исходного графика и прогноза

yy = [np.dot(w, x) for x in X]

# Построение графиков ----------------------------------------------------------

polystr = *'-0.1x⁵ + 5x⁴ - 700x²'*

plt.suptitle(*f'Задача регрессии\n (вариант №5), полином {polystr}, где x ∈ [0;10; 0,1] \n\nС L2 регуляризатором\n'*, fontsize=12)

plt.plot(x, yy, color=*'blue'*, label=*f"Прогноз, по выборке (четные) с L2"*) # Прогноз, обученный по выборке (четные) с L2 регуляризацией

plt.plot(x, y, color=*'red'*, label=*f"Исходные значения всех точек"*) # Исходные значения всех точек

plt.tick\_params(labelcolor=*'indigo'*)

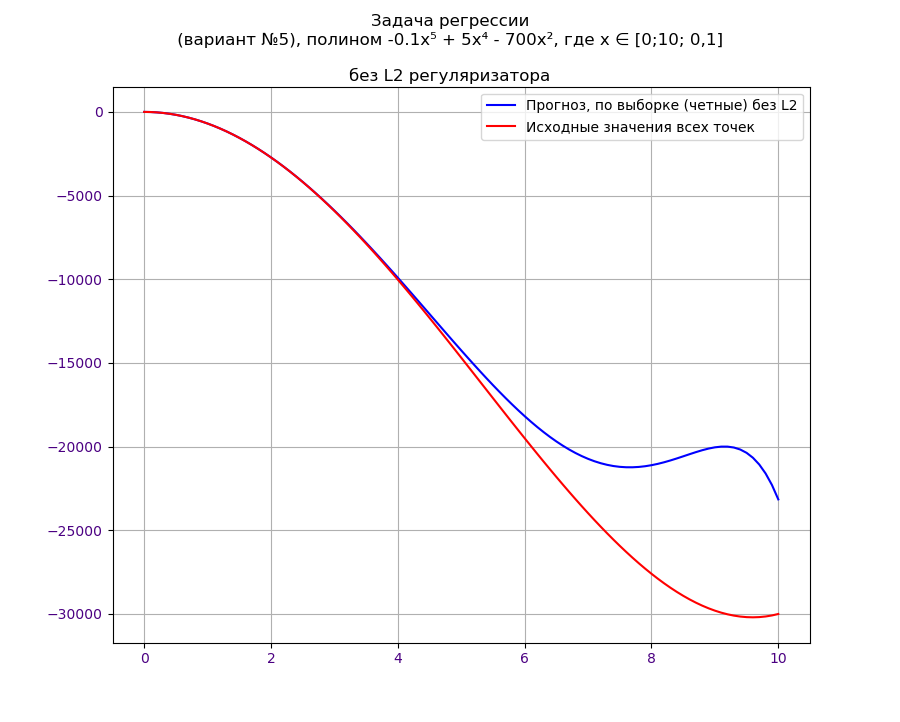
plt.legend()

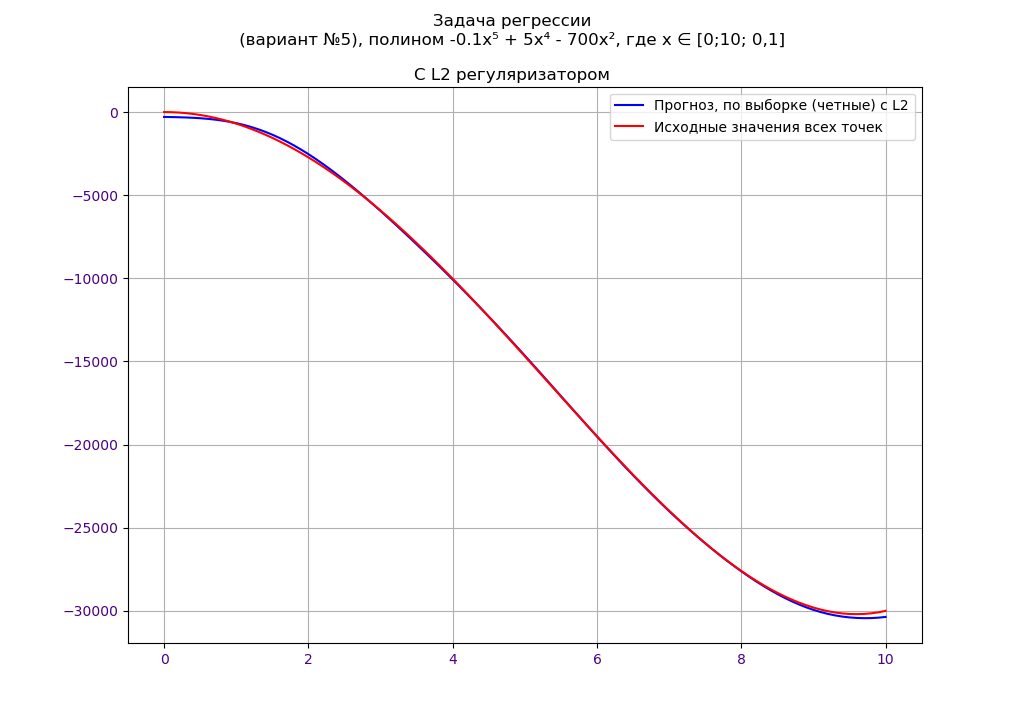
plt.grid(True)

plt.show()

Листинг 1. Код программы main.py

Результат работы программы приведен на Рисунке 1. и Рисунке 2

Рисунок 1. Результаты работы программы (без L2)

Рисунок 2. Результаты работы программы (с L2)

Коэффициенты полинома:

[ 1.22676961e-17 -7.68731204e-16 2.12636463e-14 -3.41472012e-13

3.52199767e-12 -2.43680537e-11 1.14648702e-10 -1.00000000e-01

5.00000000e+00 -9.60285428e-10 -7.00000000e+02 -1.85271632e-10

8.34173493e-12]

Исходные: y= [0.0, -6.9995, -27.99203, '...', -30148.79997, -30087.1, -30000.0]

Предсказание по 12-полиноминальной модели: y= [0.0, -6.9995, -27.99203, '...', -30148.79997, -30087.1, -30000.0]

Весовые коэффициенты (отрегуляризированные):

[-2.93170465e+02 -6.06471729e+01 -9.92440787e+01 -1.24848747e+02

-1.06932459e+02 -1.98584077e+01 6.22666059e+01 -2.83034337e+01

6.36956209e+00 -8.28126788e-01 6.33678427e-02 -2.65910680e-03

4.73310356e-05]

# Вывод

В ходе лабораторной работы я:

1. аппроксимировал функцию своего варианта с помощью заданной линейной модели;
2. вычислил значения коэффициентов вектора для квадратической функции потерь, которые минимизируют эмпирический риск;
3. вычислил прогнозы функции с помощью полученной модели для всего диапазона значений;
4. вычислил коэффициенты вектора ;
5. для новой модели повторил вычисление прогнозов функции для всего диапазона значений.