**Федеральное государственное образовательное**

**бюджетное учреждение**

**высшего образования**

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ**

**ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ**

**ФЕДЕРАЦИИ»**

**(Финансовый университет)**

**Факультет**

**информационных технологий и анализа больших данных**

**Направление «Прикладная математика и информатика»**

**Домашнее задание № 6**

**«Логистическая регрессия, обучение SVM при помощи прямо-двойственного метода внутренней точки»**

Студенты группы ПМ19-3

Караваев Артем Евгеньевич

Лазаренко Владлена Владимировна

Минаков Артем Дмитриевич

Пластун Екатерина Сергеевна

Голомысов Даниил Владиславович

Руководитель:

Аксенов Дмитрий Андреевич

**Москва 2022**

Оглавление.

1. Математическая модель (постановка задачи)
2. Алгоритмы
   1. Алгоритм 1
      1. Описание входных данных
      2. Описание алгоритма решения
      3. Описание выходных данных
   2. Алгоритм 2
      1. Описание входных данных
      2. Описание алгоритма решения
      3. Описание выходных данных
   3. Алгоритм 3
      1. Описание входных данных
      2. Описание алгоритма решения
      3. Описание выходных данных
3. Варианты использования системы
   1. ВИ 1
   2. ВИ 2
4. Архитектура решения
   1. Функции считывания информации
   2. Функции обработки информации
   3. Функции вывода информации
5. Тестирование
6. Выводы и заключение
7. **Постановка задачи (физическая модель)**

К нам пришел заказчик с новым проектом по определению подлинности банкноты по фотографии. Наша задача будет состоять в том, чтобы по некоторым признакам фотографии определить, является ли банкнота фальшивой или подлинной.

1. **Математическая модель и алгоритмы**

Обозначим X1, X2, X3, X4 – признаки, по которым будет определять подлинность банкноты. Х1 - Дисперсия вейвлета. Преобразованное изображение (непрерывное). Х2 - Skewness of Wavelet. Х3 - Куртоз Вейвлета. Преобразованное изображение (сплошное). Х4 - Энтропия изображения (непрерывная). Y – предсказываемая переменная (0, 1): 0 – банкнота является фальшивой, 1 – подлинной.

* 1. **Алгоритм 1**

**Модель классификации на основе логистической регрессии.**

[Логистическая регрессия](https://onthe.io/learn/en/category/qna/What-is-logistic-regression) является одним из статистических методов классификации с использованием линейного дискриминанта Фишера. Также она входит в [топ](https://onthe.io/learn/en/category/qna/What-are-the-best-algorithms-for-the-data-scientists%3F) часто используемых алгоритмов в науке о данных. В этой статье суть логистической регрессии описана так, что она станет понятна даже людям не очень близким к статистике.

* + 1. **Описание входных данных**

**Обязательные параметры:**

а) массив обучающей выборки (, , );

б) массив предсказываемой переменной ();

**Необязательные параметры:**

в) вид регуляризации (по умолчанию None, регрессия выполняется без нее);

г) построение графика классификации (по умолчанию False), определяет, будет ли построен график.

* + 1. **Описание алгоритма решения**

1. Вычислить значение  граничной функции (или, как вариант, функцию отношения шансов). Для простоты обозначим эту величину image.

2. Вычислить отношение шансов:  . (так как t является логарифмом).

3. Имея значение , вычислить  с помощью простой зависимости.

Получив значение t в шаге 1, можно объединить шаги 2 и 3:  
Правая часть уравнения, указанного выше, называется логистической функцией. Отсюда и название, данное этой модели обучения.

* + 1. **Описание выходных данных**

а) массив предсказанных классов (в формате [,]);

б) массив коэффициентов регрессии ();

в) график классификации, если стоит соответствующий параметр.

* 1. **Алгоритм 2**

**Модель классификации на основе логистической регрессии с радиальными базисными функциями.**

Такие сети позволяют нам перейти к следующему этапу использования приближений функций. Сети радиальных базисных функций можно представить двумя способами. Первый состоит в том, сеть радиальных базисных функций – это на самом деле линейная модель, в которой мы вначале произвели извлечение признаков, а признаки стали ядрами радиальных базисных функций. Вскоре мы обсудим, что такое ядра радиальных базисных функций. Второй способ представления сети радиальных базисных функций – в виде нейронной сети с одним скрытым слоем и радиальными базисными функциями в качестве функции активации.

* + 1. **Описание входных данных**

**Обязательные параметры:**

а) массив обучающей выборки (, , );

б) массив предсказываемой переменной ();

* + 1. **Описание алгоритма решения**

1. Выбрать несколько точек из пространства состояний. Это позволяет ограничить количество используемых экземпляров. Не забывайте, что команда.

2. Подставляем признаки в модель линейной регрессии и используем градиентный спуск для обновления параметров модели линейной регрессии.

3. В отличие от нейронной сети прямого распространения, признаки не изменяются по мере обучения – экземпляры, выбранные вначале, остаются теми же навсегда. Хотя это может показаться ограничением, но на самом деле оно работает лучше, чем обычная нейронная сеть прямого распространения, где мы стараемся запустить стохастическое обратное распространение через всю нейронную сеть.

* + 1. **Описание выходных данных**

а) массив предсказанных классов (в формате [,]);

б) массив коэффициентов регрессии ();

в) график классификации, если стоит соответствующий параметр.

* 1. **Алгоритм 3**

**Модель классификации на основе логистической регрессии с регуляризацией L1.**

Методы регуляризации используются для уменьшения погрешности путем соответствующей подгонки функции к данному тренировочному набору, чтобы избежать переобучения. Эти функции существенно уменьшают коэффициенты (β) каждого признака, тем самым уменьшая шансы на отмену значений.

* + 1. **Описание входных данных**

**Обязательные параметры:**

а) массив обучающей выборки (, );

б) массив предсказываемой переменной ();

**Необязательные параметры:**

в) вид регуляризации (по умолчанию None, регрессия выполняется без нее);

г) построение графика классификации (по умолчанию False), определяет, будет ли построен график.

* + 1. **Описание алгоритма решения**

В логистической регрессии использование -регуляризации вместо квадратичной соответствует решению следующей задачи оптимизации на этапе обучения:

\frac{1}{N}\sum_{n=1}^N\log(1+\exp(-t_n\vec{w}^T\vec{\phi}(\vec{x}_n))) + \lambda\sum_{j=1}^D|w_j|\rightarrow\min_{\vec{w}}.

Известно, что решение такой задачи обладает свойством разреженности, т.е. часть компонент оптимального вектора весов \vec{w} равно нулю. С точки зрения линейного решающего правила \mathrm{sign}(\vec{w}^T\vec{\phi}(\vec{x})) нулевые веса равносильны исключению соответствующей базисной функции (или исходного признака) из модели.

* + 1. **Описание выходных данных**

а) массив предсказанных классов (в формате [,]);

б) массив коэффициентов регрессии ();

в) график классификации, если стоит соответствующий параметр.

* 1. **Алгоритм 4**

**Модель классификации на основе метода опорных векторов.**

Суть метода заключается в выборе прямой, расстояние от которой до каждого класса максимально. Другими словами, выбирается та прямая, которая разделяет классы наилучшим образом. Вектора, лежащие ближе всех к разделяющей гиперплоскости, называются *опорными векторами* (support vectors).

**Описание входных данных**

**Обязательные параметры:**

а) массив обучающей выборки (, );

б) массив предсказываемой переменной ();

**Необязательные параметры:**

в) вид регуляризации (по умолчанию None, регрессия выполняется без нее);

г) построение графика классификации (по умолчанию False), определяет, будет ли построен график.

* + 1. **Описание алгоритма решения**

1. Пусть имеется обучающая выборка:   
2. Метод опорных векторов строит классифицирующую функцию *F* в виде , где  — скалярное произведение, ***w*** — нормальный вектор к разделяющей гиперплоскости, *b* — вспомогательный параметр.

3. Те объекты, для которых *F(****x****) = 1* попадают в один класс, а объекты с*F(****x****) = -1* попадают в другой. Выбор именно такой функции неслучаен: любая гиперплоскость может быть задана в виде  для некоторых ***w*** и *b.*

*4.* Далее выбираются такие ***w*** и *b* которые максимизируют расстояние до каждого класса. Можно подсчитать, что данное расстояние равно . Проблема нахождения максимума  эквивалентна проблеме нахождения минимума .

5. Нахождение максимума   является стандартной задачей квадратичного программирования и решается с помощью множителей Лагранжа.

* + 1. **Описание выходных данных**

а) массив предсказанных классов (в формате [,]);

б) массив коэффициентов регрессии ();

в) график классификации, если стоит соответствующий параметр.

1. **Варианты использования системы**

Предполагается, что пользователь будет предлагать программе на вход файл формата csv, excel, txt. Функция сама будет определять тип файл и преобразовывать его, вследствие чего выявлять сама по названию столбцов переменные У и Х.

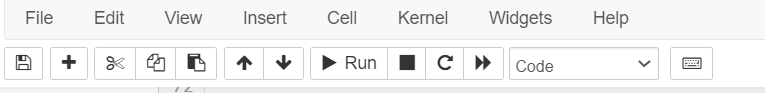
В соответствии с нужной моделью классификации пользователь будет выбирать нужную ему функцию, так как они будут иметь говорящие названия.

На выходе пользователь должен получить коэффициенты регрессии полученной модели, а также массив предсказанных классов. По желанию, пользователю можно представить график классификации, если стоит соответствующий параметр.

* 1. **ВИ** 
     1. **Ввод данных.**
        1. **Ввод переменных Х и У.**

Вы можете ввести любые две английские строчные или прописные буквы через запятую, после чего нажмите «Run». После ввода переменных пользователь получит результаты работы функции. Пример:







* + - 1. **Вывод полученных результатов.**

В первую очередь пользователь увидит метрики качества модели, такие как Accuracy, Precision и Recall.

**Изображение выглядит как текст

Автоматически созданное описание**

**Первая строка** выводит метрику Accuracy - доля правильных ответов алгоритма.

**Вторая** строка вывод значение метрики Precision - доля объектов, названных классификатором положительными и при этом действительно являющимися положительными

**Третья** строка вывод значение метрики Recall, которая показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм.

Следующей строкой пользователь увидит полученные коэффициенты регрессии. Каждое число в массиве соответствует коэффициенту при конкретном Хi.

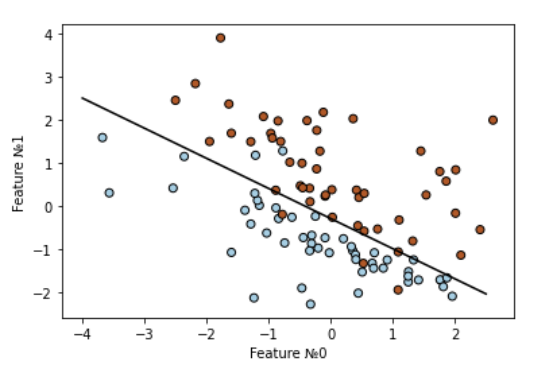


Далее пользователю представится возможность увидеть словарь предсказанных значений, то есть к какому классу была отнесен каждый X. Ключ словаря – список Xit, значение ключа – число 0 или 1 (класс, к которому был отнесен список Xit.

Изображение выглядит как текст

Автоматически созданное описание

Пользователю также предоставляется возможность увидеть результат работы алгоритма на графике. Каждый класс соответствует определенному цвету.



1. **Архитектура решения**
   1. **Метод классификации на основе логистической регресии.**
      1. **Функция считывания, обработки и вывода информации.**

**def logistic():**

"""

Функция, реализующая модель классификации на два класса основе логистической регрессии.

Parameters

----------

x\_, y\_: numpy array

Массивы обучающей переменной и предсказываемой

x\_train, x\_test, y\_train, y\_test: numpy array

Массивы обучающей переменной и предсказываемой, разделенные на обучающую и тестовую выборки

y\_pred: numpy array

Массив предсказанных значений

Returns

-----------

Accuracy, precision, recall: float

Метрики точности модели

cls.coef\_: numpy array

Массив с коэффициентами модели классификации

dicc: dict

Словарь вида {x\_test: y\_pred}, тестовое значение переменной х сопоставляется с предсказанным значением переменной y

"""

* 1. **Метод классификации на основе логистической регрессии с радиальной базисной функцией.**
     1. **Функция считывания, обработки и вывода информации.**

**def logistic\_rbf():**

"""

Функция, реализующая модель классификации на два класса основе логистической регрессии c радиальной базисной функцией.

Parameters

----------

x\_, y\_: numpy array

Массивы обучающей переменной и предсказываемой

x\_train, x\_test, y\_train, y\_test: numpy array

Массивы обучающей переменной и предсказываемой, разделенные на обучающую и тестовую выборки

y\_pred: numpy array

Массив предсказанных значений

Returns

-----------

Accuracy, precision, recall: float

Метрики точности модели

cls.coef\_: numpy array

Массив с коэффициентами модели классификации

dicc: dict

Словарь вида {x\_test: y\_pred}, тестовое значение переменной х сопоставляется с предсказанным значением переменной y

"""

* + 1. Модель классификации на основе логистической регрессии c радиальной базисной функцией.

**Функция считывания, обработки и вывода информации.**

**def logistic\_rbf():**

"""

Функция, реализующая модель классификации на два класса основе логистической регрессии c радиальной базисной функцией.

Parameters

----------

x\_, y\_: numpy array

Массивы обучающей переменной и предсказываемой

x\_train, x\_test, y\_train, y\_test: numpy array

Массивы обучающей переменной и предсказываемой, разделенные на обучающую и тестовую выборки

y\_pred: numpy array

Массив предсказанных значений

Returns

-----------

Accuracy, precision, recall: float

Метрики точности модели

cls.coef\_: numpy array

Массив с коэффициентами модели классификации

dicc: dict

Словарь вида {x\_test: y\_pred}, тестовое значение переменной х сопоставляется с предсказанным значением переменной y

"""

* 1. **Модель классификации на основе логистической регрессии с регуляризацией L1.**

**def lin\_with\_l1() – аналогично описанному в 4.2.**

* 1. **Модель классификации на основе метода опорных векторов.**

**def vvod():**

"""

Функция для ввода данных, с помощью которых будет проводиться классификация

Parameters

----------

X: numpy.ndarray

Массив значений переменной x

y: numpy.ndarray

Массив значений переменной y

X\_train: numpy.ndarray

Массив обучающей выборки переменной x

X\_test: numpy.ndarray

Массив тестовой выборки переменной x

y\_train: numpy.ndarray

Массив обучающей выборки переменной y

y\_test: numpy.ndarray

Массив тестовой выборки переменной y

Returns

-----------

Final: dictionary

Словарь со всеми переменными, которые ранее были введены, нужен для ввода в основную функцию

"""

def SVM(dictionary):

"""

Функция, реализующая модель классификации на два класса на основе метода опорных векторов.

Parameters

----------

dictionary: dictionary

Словарь со всеми переменными, которые ранее были введены в функции vvod

Returns

-----------

dicc: dict

Словарь вида {x\_test: y\_pred}, тестовое значение переменной х сопоставляется с предсказанным значением переменной y

"""

1. **Тестирование**

Таблица 1. Результаты тестирования моделей классификации

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Входные данные** | **Логистическая** | **Логистическая с радиальной** | **Логистическая с регуляризацей** | **Метод опорных векторов** |
| X1, Y1 | Коэффициенты регрессии: [[1.52107931 0.04507077 0.62042649 1.11848963]]  Accuracy: 0,84 | Accuracy: 0,86 | Коэффициенты регрессии: [[2.59002227 0. 0.04899888 0. ]]  Accuracy: 0,84 | Accuracy: 0,86 |
| X2, Y2 | Коэффициенты регрессии: [[-0.07974966 -0.31857647 0.72713819 -0.17202434]]  Accuracy: 0,7125 | Accuracy: 0,7685 | Коэффициенты регрессии: [[-0.0127285 -0.537696 0.67852901 0. ]]  Accuracy: 0,7125 | Accuracy: 0,7685 |
| X3, Y3 | Коэффициенты регрессии: [[-0.46960565 -1.50083151 2.04414003 -1.03981815]]  Accuracy: 0,865 | Accuracy: 0,891 | Коэффициенты регрессии: [[-0.04712968 0. 2.86416806 -1.80319369]]  Accuracy: 0,865 | Accuracy: 0,891 |
| X4, Y4 | Коэффициенты регрессии: [[ 0.87314589 1.05027479 -0.06913807 0.31028784]]  Accuracy: 0,778 | Accuracy: 0,824 | Коэффициенты регрессии: [[0.3553392 1.57685332 0. 0. ]]  Accuracy: 0,778 | Accuracy: 0,814 |
| X5, Y5 | Коэффициенты регрессии: [[ 0.45488621 -0.09449011 0.4332556 -0.09722093]]  Accuracy: 0,68 | Accuracy: 0,76 | Коэффициенты регрессии: [[0.71897912 0. 0.18935884 0. ]]  Accuracy: 0,68 | Accuracy: 0,76 |

Таблица 2. Время выполнения в секундах.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Входные данные** | **Логистическая** | **Логистическая с радиальной** | **Логистическая с регуляризацей** | **Метод опорных векторов** |
| X1, Y1 | 0,08 | 0,052 | 0,04 | 0,052 |
| X2, Y2 | 0,168 | 2,099 | 0,143 | 2,07 |
| X3, Y3 | 0,1 | 0,339 | 0,113 | 0,32 |
| X4, Y4 | 0,06 | 0,125 | 0,056 | 0,11 |
| X5, Y5 | 0,04 | 0,029 | 0,03 | 0,033 |

По результатам вычисления скорости, самый эффективный метод для использования – логистическая регрессия с регуляризацией L1.

1. **Заключение**

**Решение задачи заказчика:**

**Классификация на основе логистической регрессии:**

Coef = [[-3.14111688 -1.7986436 -2.1781067 -0.15452799]]

Accuracy = 0.9927272727272727

Precision = 0.9916666666666667

Recall = 0.9936305732484076

**Классификация на основе логистической регрессии c радиальной базисной функцией:**

Accuracy = 1.0

Precision = 1.0

Recall = 1.0

**Классификация на основе логистической регрессии c регуляризацией L1:**

Accuracy = 0.9927272727272727

Precision = 0.9916666666666667

Recall = 0.9936305732484076

Коэффициенты регрессии: [[-3.89357489 -2.16976984 -2.65955853

- .25023214]]

**Классификация на основе метода опорных векторов:**

Accuracy = 0.9927272727272727

Precision = 0.9916666666666667

Recall = 0.9936305732484076

Коэффициенты регрессии: [[-2.49795915 -1.44684902 -1.73069754 -0.27196743]]

Все написанные алгоритмы дают высокую предсказательную точность, однако наиболее точной является классификация на основе логистической регрессии c радиальной базисной функцией.