**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**

«Классификация»

Студенты группы ПМ19-3:

Захаров Д. В.

Исмоилова М. В.

Константинов К. Л.

Мосолова К. Д.

Самофалова Т. А.

Руководитель:

Аксенов Дмитрий Андреевич

**Москва 2022**

**Содержание**

[1. Постановка задачи (физическая модель) 3](#_Toc105863733)

[2. Математическая модель и алгоритмы 3](#_Toc105863734)

[4. Варианты использования системы 8](#_Toc105863735)

[5. Архитектура решения. 11](#_Toc105863736)

[6. Тестирование 14](#_Toc105863737)

[Заключение 16](#_Toc105863738)

# **Постановка задачи (физическая модель)**

Новостному агентству «Рога и копыта» необходимо проставить метки на хранящиеся в архиве материалы. Новостей достаточно много для ручной обработки. Сотрудники агентства определили классы и предоставили начальный размеченный датасет. Для решения задачи классификации было решено использовать следующие алгоритмы:

1. Функцию, реализующую модель классификации на два класса на основе логистической регрессии.
2. Функцию, реализующую модель классификации на два класса на основе логистической регрессии с радиальными базисными функциями.
3. Функцию, реализующую модель классификации на два класса на основе логистической регрессии с регуляризацией L1.
4. Функцию, реализующую модель классификации на два класса на основе метода опорных векторов.
5. Функцию, реализующую модель классификации на два класса с применением прямо-двойственного метода внутренней точки для задачи обучения метода опорных векторов.

Под каждый обозначенный выше алгоритм была написана отдельная функция (их описание см. в разделе 5). Также была сделана визуализация работы логистической регрессии с l1, замерена скорость работы каждого алгоритма и проведено тестирование на сгенерированных и реальных данных. Алгоритмы были сравнены между собой, был рекомендован лучший алгоритм и предложено дальнейшее усовершенствование функции (более подробное описание каждого действия см. в следующих разделах).

# **Математическая модель и алгоритмы**

Рассмотрим задачу классификации на два класса. Имеется обучающая выборка , где – вектор признаков для объекта n, а – его метка класса. Задача заключается в предсказании метки класса для объекта, представленного своим вектором признаков . (1)

**Логистическая регрессия**

В *логистической регрессии* предсказание метки класса осуществляется по знаку линейной функции:

), (2) где w- вектор весов.

Настройка весов осуществляется путем минимизации функции потерей:

(3)

Как правило, для нахождения минимума применяют метод Ньютона.

Для снижения вероятности переобучения алгоритма как правило используют *регуляризацию*. Используемый для этого параметр С отвечает за силу регуляризации; чем меньше значение параметра, тем сильнее будет регуляризация.

Функция потерь для регуляризации *l2* имеет вид:

(4)

Функция потерь для *l1*:

(5)

*Elastic-Net* является комбинацией l2 и l1. Функция потерь:

(6)

Параметр ρ в Elastic-Net контролирует силу l1-регуляризации: если ρ = 1, то (6) преобразуется в (5); если ρ = 0, то (6) преобразуется в (4).

В формулировке задачи (1) говорится о классификации на два класса, однако все сказанное выше можно применить и для классификации на большее количество классов y={0, 1, …, m}. В этом случае мы делим задачу на m + 1 двоичных задач классификации; в каждом из них мы прогнозируем вероятность того, что 'y' является членом одного из наших классов. Мы в основном выбираем один класс, а затем объединяем всех остальных в один второй класс. Мы делаем это неоднократно, применяя двоичную логистическую регрессию к каждому случаю, а затем используем гипотезу, которая вернула наивысшее значение в качестве нашего прогноза.

Поскольку возможны ситуации, когда классы не могут быть линейно разделимы друг от друга, в качестве базы для логистической регрессии предпочтительнее использовать не линейную регрессию, а полиномиальную, имеющую вид:

(7).

**Радиальные базисные функции**

Одним их стандартных способов обобщения логистической регрессии на случай построения нелинейных разделяющих поверхностей является использование радиальных базисных функций (Radial Basis Function, RBF). Тогда (2) преобразуется в:

)), (8)

где – j-й объект обучающей выборки,

**Метод опорных векторов (SVM)**

Основная идея метода заключается в построении гиперплоскости, разделяющей объекты выборки оптимальным способом. Алгоритм работает в предположении, что чем больше расстояние (зазор) между разделяющей гиперплоскостью и объектами разделяемых классов, тем меньше будет средняя ошибка классификатора.

В пространстве уравнение при заданных и b определяет гиперплоскость — множество векторов, принадлежащих пространству меньшей размерности . Данная гиперплоскость делит пространство на два класса.

Пусть выборка линейно разделима, то есть существует некоторая гиперплоскость, разделяющая классы −1 и +1. Тогда в качестве алгоритма классификации можно использовать линейный пороговый классификатор:

(9), где и b – параметры гиперплоскости.

Введем понятие отступа. Отступ - характеристика, оценивающая, насколько объект "погружён" в свой класс, насколько типичным представителем класса он является. Чем меньше значение отступа , тем ближе объект подходит к границе классов и тем выше становится вероятность ошибки. Отступ отрицателен тогда и только тогда, когда алгоритм допускает ошибку на объекте .

Если выборка линейно разделима, то существует такая гиперплоскость, отступ от которой до каждого объекта положителен:

(10)

Таким образом, получаем задачу:

(11)

На практике линейно разделимые выборки практически не встречаются: в данных возможны выбросы и нечёткие границы между классами. В таком случае поставленная выше задача не имеет решений, и необходимо ослабить ограничения, позволив некоторым объектам попадать на "территорию" другого класса. Для каждого объекта отнимем от отступа некоторую положительную величину , но потребуем, чтобы эти введённые поправки были минимальны. Это приведёт к следующей постановке задачи:

(12)

Как и ранее, параметр C здесь отвечает за силу регуляризации.

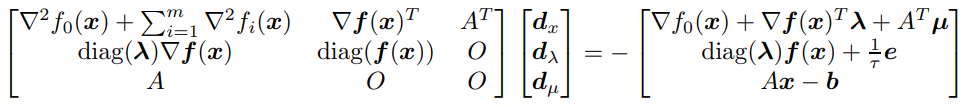
Задачу (12) можно переписать в более привычном виде:

(13)

Для задачи обучения метода опорных векторов зачастую используют прямо-двойственный метод внутренней точки.

Прямо-двойственный метод Ньютона оптимизирует одновременно прямые переменные x и двойственные переменные µ путем решения линеаризованной системы Куна-Таккера:

(14)

Направления оптимизации находятся путем решения следующей СЛАУ (15):

Здесь  **–** это вектор-столбец ограничений типа неравенство; – это матрица, на диагонали которой расположен вектор (размера m\*1); e – единичный вектор; А – матрица коэффициентов в ограничении типа равенств, где на позиции (i,j ) расположен коэффициент в j ограничении при i-ой переменной.

Если обозначить левую матрицу как M1, а правую как M2, то = *(16)*

Вычислив направления оптимизации, можно вычислить и длину шага:

*(17)*

находим с помощью метода поиска backtracking: выбираем как элемент последовательности {1, ½, ¼ ,…}, пока не найдем то, при котором будет выполняться ).

После чего выполняется обновление значений:

*(18)*

*(19)*

*(20)*

Итерации по нахождению новой точки проводятся, пока не выполнится условие

где e = 10^(-6).

**3. Алгоритмы**

**3.1. Логистическая регрессия**

1. Массивы и преобразуются к виду (7), где n-заданная пользователем степень полинома.

2. Веса w находятся путем решения задачи (3), (4), (5) или (6) в зависимости от указанной пользователем регуляризации.

3. С помощью найденных весов по формуле (2) предсказывается вектор .

**3.2. Логистическая регрессия с радиальными базисными функциями**

1. Веса w находятся путем решения задачи (3), (4), (5) или (6) в зависимости от указанной пользователем регуляризации.

2. С помощью найденных весов по формуле (8) предсказывается вектор .

**3.3. Логистическая регрессия с регуляризацией L1**

1. Массивы и преобразуются к виду (7), где n-заданная пользователем степень полинома.

2. Веса w находятся путем решения задачи (5), если вид регуляризации не был указан, и путем решения задачи (6), если указан вид регуляризации l2 или Elastic-Net.

3. С помощью найденных весов по формуле (2) предсказывается вектор .

**3.4 Метод опорных векторов**

1. Веса w находятся путем решения задачи (12).

2. С помощью найденных весов по формуле (9) предсказывается вектор .

**3.5. Прямо-двойственный метод внутренней точки для задачи обучения метода опорных векторов.**

1. Находятся направления оптимизации по формуле (15).

2. Вычисляется длина шага по формуле (11) описанному выше методу backtracking.

3. Вычисляются новые значения по формулам (18-20).

4. Проверяется выполнение условие (21). Если оно выполняется, алгоритм прекращает свою работу, в противном случае повторяются шаги 2-4.

5. Найденные на предыдущем шагу веса используются для предсказания вектора по формуле (9).

# **4. Варианты использования системы**

ВИ1: пользователь хочет решить задачу при помощи функции, реализующей модель классификации на два класса на основе логистической регрессии.

1. Пользователь применяет функцию own\_logistic\_regression() В качестве обязательных параметров передаются:
2. X\_train (np\_array() признаков для прогнозирования)
3. y\_train (np.array() значений классов для тренировочной выборки)
4. X\_test (np.array() значений признаков для прогнозирования
5. Если пользователь хочет изменить вид регуляризации (по умолчанию none), он может изменить значение параметра penalty
6. Пользователь может варьировать силу регуляризации при помощи параметра C (по умолчанию 1, значения должны быть больше 0)
7. Пользователь может изменять значение параметра l1\_ratio, если в качестве вида регуляризации выбран 'elasticnet'. (по умолчанию ‘none’ (может быть только 0<=l1\_ratio<=1))
8. Если пользователь хочет изменить степень используемого полинома, он может варьировать параметр degree (по умолчанию 1, может быть только целым положительным числом)
9. для изменения максимального кол-ва итераций, пользователь может менять значение параметра max\_iter (по умолчанию 100)
10. Если пользователю необходимо визуализировать полученную регрессию, он меняет значение параметра draw на True (по умолчанию False)
11. В качестве выходных параметров пользователь получает:
12. Кортеж из двух элементов;: np.array() предсказанных классов и np.array() коэффициентов регрессии;
13. В случае, если draw = True, строится график построенной регрессии

ВИ2: пользователь хочет решить задачу при помощи функции, реализующей модель классификации на два класса на основе логистической регрессии с радиальными базисными функциями.

1. Пользователь применяет функцию logistic\_regression\_with\_rbf() В качестве обязательных параметров передаются:
2. X\_train (np\_array() признаков для прогнозирования)
3. y\_train (np.array() значений классов для тренировочной выборки)
4. X\_test (np.array() значений признаков для прогнозирования
5. Пользователь может варьировать силу регуляризации при помощи параметра C (по умолчанию 1, значения должны быть больше 0)
6. Пользователь может изменять коэффициент ядра, варьируя параметр gamma (по умолчанию ‘scale’, может быть только gamma>0)
7. для изменения максимального кол-ва итераций, пользователь может менять значение параметра max\_iter (по умолчанию 100)
8. Если пользователю необходимо визуализировать полученную регрессию, он меняет значение параметра draw на True (по умолчанию False)
9. В качестве выходных параметров пользователь получает:
10. Кортеж из двух элементов;: np.array() предсказанных классов и np.array() коэффициентов регрессии;
11. В случае, если draw = True, строится график построенной регрессии.

ВИ3: пользователь хочет решить задачу при помощи функции, реализующей модель классификации на два класса на основе логистической регрессии с регуляризацией L1.

1. Пользователь применяет функцию logistic\_regression\_with\_l1() В качестве обязательных параметров передаются:
2. X\_train (np\_array() признаков для прогнозирования)
3. y\_train (np.array() значений классов для тренировочной выборки)
4. X\_test (np.array() значений признаков для прогнозирования
5. Если пользователь хочет изменить вид регуляризации (по умолчанию none), он может изменить значение параметра penalty
6. Пользователь может варьировать силу регуляризации при помощи параметра C (по умолчанию 1, значения должны быть больше 0)
7. Пользователь может изменять значение параметра l1\_ratio, если в качестве вида регуляризации выбран 'elasticnet'. (по умолчанию ‘none’ (может быть только 0<=l1\_ratio<=1))
8. Если пользователь хочет изменить степень используемого полинома, он может варьировать параметр degree (по умолчанию 1, может быть только целым положительным числом)
9. для изменения максимального кол-ва итераций, пользователь может менять значение параметра max\_iter (по умолчанию 100)
10. Если пользователю необходимо визуализировать полученную регрессию, он меняет значение параметра draw на True (по умолчанию False)
11. В качестве выходных параметров пользователь получает:
12. Кортеж из двух элементов;: np.array() предсказанных классов и np.array() коэффициентов регрессии;
13. В случае, если draw = True, строится график построенной регрессии

ВИ4: пользователь хочет решить задачу при помощи функции, реализующей модель классификации на два класса на основе метода опорных векторов.

1. Пользователь применяет функцию own\_svm() В качестве обязательных параметров передаются:
2. X\_train (np\_array() признаков для прогнозирования)
3. y\_train (np.array() значений классов для тренировочной выборки)
4. X\_test (np.array() значений признаков для прогнозирования
5. Если пользователь хочет изменить вид регуляризации (по умолчанию none), он может изменить значение параметра penalty
6. Пользователь может варьировать силу регуляризации при помощи параметра C (по умолчанию 1, значения должны быть больше 0)
7. для изменения максимального кол-ва итераций, пользователь может менять значение параметра max\_iter (по умолчанию 100)
8. Если пользователю необходимо визуализировать полученную регрессию, он меняет значение параметра draw на True (по умолчанию False)
9. В качестве выходных параметров пользователь получает:
10. Кортеж из двух элементов;: np.array() предсказанных классов и np.array() коэффициентов регрессии;
11. В случае, если draw = True, строится график построенной регрессии

ВИ5: пользователь хочет решить задачу при помощи функции, реализующей модель классификации на два класса с применением прямо-двойственного метода внутренней точки для задачи обучения метода опорных векторов

1. Пользователь применяет функцию svm\_with\_pddipm() В качестве обязательных параметров передаются:
2. X\_train (np\_array() признаков для прогнозирования)
3. y\_train (np.array() значений классов для тренировочной выборки)
4. X\_test (np.array() значений признаков для прогнозирования
5. Если пользователь хочет изменить вид регуляризации (по умолчанию none), он может изменить значение параметра penalty
6. Пользователь может варьировать силу регуляризации при помощи параметра C (по умолчанию 1, значения должны быть больше 0)
7. для изменения максимального кол-ва итераций, пользователь может менять значение параметра max\_iter (по умолчанию 100)
8. Если пользователю необходимо визуализировать полученную регрессию, он меняет значение параметра draw на True (по умолчанию False)
9. В качестве выходных параметров пользователь получает:
10. Кортеж из двух элементов;: np.array() предсказанных классов и np.array() коэффициентов регрессии;
11. В случае, если draw = True, строится график построенной регрессии

# **5. Архитектура решения.**

own\_logistic\_regression ­– Функция, реализующая двухклассовую модель классификации на основе логистической регрессии.

Параметры

----------

X\_train : numpy.array

Набор признаков для обучающей выборки.

y\_train : numpy.array

Набор значений классов для обучающей выборки.

X\_test : numpy.array

Набор признаков для тестовой выборки.

penalty : string, default='none'

Тип регуляризации. ['l1', 'l2', 'elasticnet', 'none'].

C : float, по умолчанию=1

Сила регуляризации.

l1\_ratio : float, default='none'

Сила l1- регуляризации, если выбрано значение penalty=’elasticnet’.

degree : int, по умолчанию=1

Степень полинома.

draw : int, по умолчанию=False

Построение графика классификации.

max\_iter : int, по умолчанию=100

Количество итераций обучения

logistic\_regression\_with\_rbf – функция, реализующая двухклассовую модель классификации на основе логистической регрессии с радиальными базисными функциями.

Параметры

----------

X\_train : numpy.array

Набор признаков для обучающей выборки.

y\_train : numpy.array

Набор значений классов для обучающей выборки.

X\_test : numpy.array

Набор признаков для тестовой выборки.

C : float, по умолчанию=1

Сила регуляризации.

gamma : float, по умолчанию='scale'.

Коэффициент ядра.

draw : int, по умолчанию=False

Построение графика классификации.

max\_iter : int, по умолчанию=100

Количество итераций обучения

logistic\_regression\_with\_l1 – Функция, реализующая двухклассовую модель классификации на основе линейной регрессии.

Параметры

----------

x\_train : numpy.array

Набор признаков для обучающей выборки.

y\_train : numpy.array

Набор значений классов для обучающей выборки.

x\_test : numpy.array

Набор признаков для тестовой выборки.

penalty : string, default='none'

Тип регуляризации. ['l1', 'l2', 'elasticnet', 'none'].

C : float, по умолчанию=1

Сила регуляризации.

l1\_ratio : float, default='none'

Сила l1- регуляризации, если выбрано значение penalty=’elasticnet’.

degree : int, по умолчанию=1

Степень полинома.

draw : int, по умолчанию=False

Построение графика классификации.

own\_svm – функция, реализующая двухклассовую модель классификации на основе методы опорных векторов

Параметры

----------

x\_train: numpy.array

Набор признаков для обучающей выборки.

y\_train: numpy.array

Набор значений классов для обучающей выборки.

X\_test: numpy.array

Набор признаков для тестовой выборки.

C: float, по умолчанию=1

Сила регуляризации.

penalty: string, default='none'

Тип регуляризации. ['l1', 'l2', 'elasticnet', 'none'].

graph: int, по умолчанию=False

Построение графика классификации.

svm\_with\_pddipm – функция, реализующая двухклассовую модель классификации с использованием прямо-двойственногометодавнутренней точки для задачи обучения методом опорных векторов

Параметры

----------

X\_train : numpy.array

Набор признаков для обучающей выборки.

y\_train : numpy.array

Набор значений классов для обучающей выборки.

X\_test : numpy.array

Набор признаков для тестовой выборки.

penalty : string, default='none'

Тип регуляризации. ['l2', 'none'].

C : float, по умолчанию=1

Сила регуляризации.

draw : int, по умолчанию=False

Построение графика классификации.

max\_iter : int, по умолчанию=100

Количество итераций обучения

# **6. Тестирование**

Было протестировано 5 функций на 3-х наборах данных.

* 1 набор – искусственно сгенерированные данные, состоящие из 10000 точек, всего 2 признака для каждого наблюдения, количество сгенерированных центров кластеров = 2.
* 2 набор - искусственно сгенерированные данные, состоящие из 100000 точек, всего 2 признака для каждого наблюдения, количество сгенерированных центров кластеров = 6.
* 3 набор – реальные данные, состоящие из 11314 наблюдений, всего 130107 признаков для каждого наблюдения, количество предусмотренных кластеров = 20.

Качество работы алгоритмов замеряется с помощью метрик precision, recall, accuracy, кроме того, была замерена скорость работы.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Method | recall\_test1 | recall\_test2 | recall\_test3 |
| 0 | own\_logistic\_regression | 0.978351 | 0.50430 | 0.894830 |
| 1 | logistic\_regression\_with\_rbf | 0.445361 | 0.24365 | 0.812196 |
| 2 | svm\_with\_pddipm | 0.982474 | 0.41240 | 0.898365 |
| 3 | Сlassification\_linear\_regression\_l1 | 0.978351 | 0.50430 | 0.746354 |
| 4 | svm | 0.979381 | 0.48630 | 0.897923 |
|  | Method | accuracy\_test1 | accuracy\_test2 | accuracy\_test3 |
| 0 | own\_logistic\_regression | 0.9820 | 0.50430 | 0.894830 |
| 1 | logistic\_regression\_with\_rbf | 0.6085 | 0.24365 | 0.812196 |
| 2 | svm\_with\_pddipm | 0.9840 | 0.41240 | 0.898365 |
| 3 | Сlassification\_linear\_regression\_l1 | 0.9820 | 0.50430 | 0.746354 |
| 4 | svm | 0.9825 | 0.48630 | 0.897923 |
|  | Method | precision\_test1 | precision\_test2 | precision\_test3 |
| 0 | own\_logistic\_regression | 0.984440 | 0.50430 | 0.894830 |
| 1 | logistic\_regression\_with\_rbf | 0.638109 | 0.24365 | 0.812196 |
| 2 | svm\_with\_pddipm | 0.984504 | 0.41240 | 0.898365 |
| 3 | Сlassification\_linear\_regression\_l1 | 0.984440 | 0.50430 | 0.746354 |
| 4 | svm | 0.984456 | 0.48630 | 0.897923 |
|  | Method | time\_test1 | time\_test1 | time\_test1 |
| 0 | own\_logistic\_regression | 5.840523 | 15.165846 | 134.252441 |
| 1 | logistic\_regression\_with\_rbf | 17.448042 | 330.179411 | 87.835079 |
| 2 | svm\_with\_pddipm | 0.771831 | 10.993040 | 5.219718 |
| 3 | Сlassification\_linear\_regression\_l1 | 0.433747 | 3.239907 | 377.061752 |
| 4 | svm | 0.433747 | 3.120185 | 22.443329 |

Исходя из результатов, рекомендуемая функция для работы - функция, реализующая модель классификации на два класса с применением прямо-двойственного метода внутренней точки для задачи обучения метода опорных векторов.

# **Заключение**

Таким образом, исходя из всего вышеперечисленного, лучшей функцией, решающей поставленную задачу, является функция функция, реализующая модель классификации на два класса с применением прямо-двойственного метода внутренней точки для задачи обучения метода опорных векторов. В дальнейшем планируется пользовательское тестирование и выявление технических неполадок. Также, модель, реализованную при помощи этой функции также можно дообучать, используя размеченные ей же и проверенные данные. Данная функция позволяет решать не только задачу классификации на два класса, но и множественную классификацию, что является плюсом и уже позволяет применять ее в работе.