Работа защищена с оценкой ___
Ученый секретарь
доцент В.Д. Валединский

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В.Ломоносова

Механико-математический факультет
Кафедра вычислительной математики

Курсовая работа
студентки 510 группы
Бушуевой Натальи Сергеевны

Решение задачи классификации при поиске количественных соотношений структура-свойство с помощью метода опорных векторов на выборках химических элементов.

Solving the classification problem for searching of quantitative structure-activity relations using the support vector machine on samples of chemical compounds.

Научный руководитель,
д.ф.-м.н. Кумсков М.И.

Москва, 2021 год.

Аннотация.

В данной курсовой работе решена задача классификации при поиске количественных соотношений структура-свойство на структурах химических соединений с целью предсказания их свойств. Поиск количественных соотношений структура-свойство основан на применении методов математической статистики и машинного обучения для построения моделей, позволяющих по описанию структур химических соединений предсказывать их свойства. Описание структур химических соединений проводится векторным способом – формирование МД-матриц по графу химического соединения с помощью дескрипторов. Далее решается задача классификации с помощью построения разделяющей гиперплоскости в общем случае (на линейно неразделимой выборке). Все необходимые выборки и код программ находятся в репозитории github пользователя bush09. В последней главе описан реализованный алгоритм на языке Python 3.6 с использованием библиотек NumPy, Keras, Pandas.

Содержание

Введение.

С накоплением огромного количества экспериментальных данных по различным физико-химическим свойствам органических соединений стала актуальной задача их обобщения, классификации и статистической обработки. Решение этой задачи позволяет перейти к созданию методов прогнозирования свойств новых химических соединений, а также к созданию соединений с заданным набором свойств. В данной курсовой работе все исследования проводятся на выборке циклооксигеназа-2 (ЦОГ-2) – это ген, кодирующий действие циклооксигеназ (ЦОГ) – группа ферментов, участвующих в синтезе простаноидов. Целью курсовой работы является решение задачи классификации на выборке, представленной в виде матрицы "Молекула-Дескриптор с помощью метода опорных векторов в случае линейно неразделимой выборки с помощью подбора функции ядра.

В первой главе дано формальное определение задачи классификации, описаны особенности, возникающие при решениее задачи, а также описаны способы применнения результатов в реальном мире.

Во второй главе поставлена задаа "структура-свойство описаны возможные подходы к решению поставленной задачи поиска количественных отношений с точки зрения хемоинформатики и сама выборка со свойством, которое требуется классифицировать. Далее,были описаны различные виды дескрипторов для поиска отношений эквивалентонсти.

Третья глава посвящена постановке задачи для метода опорных векторов и описанию метода решения поиска разделяющей гипер-

плоскости в частном и общем случае, а также рассмотрен способ рещения с помощью функций ядра и спрямляющих поверхностей.

В последней главе приведены описание программной реализации метода, описаны полученные результаты, проведен их анализ.

Глава 1. Методы прогнозирования для решения задачи классификации.

Задача классификации в искусственном интеллекте и машинном обучении — это задача разделения множества наблюдений (объектов) на группы, называемые классами, на основе анализа их формального описания[4]. При классификации каждая единица наблюдения относится определенной группе или номинальной категории на основе некоторого качественного свойства.

Пусть X — множество описаний объектов, Y — конечное множество номеров (имен, меток) классов. Существует неизвестная целевая зависимость — отображение $y^* : X \to Y$, значения которой известны только на объектах конечной обучающей выборки $X^m = (x_1, y_1)(x_2, y_2)...(x_m, y_m)$. Требуется построить алгоритм $a : X \to Y$, способный классифицировать произвольный объект $x \in X$ [5].

В математической статистике задачи классификации называются также задачами дискриминантного анализа.

В машинном обучении задача классификации решается с использованием обучения с учителем, поскольку классы определяются заранее и для примеров обучающего множества метки классов заданы. Аналитические модели, решающие задачу классификации, называются классификаторами.

Задача классификации представляет собой одну из базовых задач прикладной статистики и машинного обучения, а также искусственного интеллекта в целом. Это связано с тем, что классификация является одной из наиболее понятных и простых для интерпретации технологий анализа данных, а классифицирующие

правила могут быть сформулированы на естественном языке.

К числу распространенных методов решения задачи классификации относятся:

- нейронные сети;
- логистическая и пробит-регрессия;
- деревья решений;
- метод ближайшего соседа;
- машины опорных векторов;
- дискриминантный анализ.

Задача классификации применяется во многих областях:

- в торговле — классификация клиентов и товаров позволяет оптимизировать маркетинговые стратегии, стимулировать продажи, сокращать издержки;
- в сфере телекоммуникаций — классификация абонентов позволяет определять уровень лояльности, разрабатывать программы лояльности;
- в медицине и здравоохранении — диагностика заболеваний, классификация населения по группам риска;
- в банковской сфере — кредитный скоринг.

Методы решения.

- Байесовский классификатор

  Класс алгоритмов классификации, основанный на принципе максимума апостериорной вероятности. Для классифицируемого объекта вычисляются функции правдоподобия каждого

из классов, по ним вычисляются апостериорные вероятности классов. Объект относится к тому классу, для которого апостериорная вероятность максимальна.

Байесовский подход к классификации основан на теореме, утверждающей, что если плотности распределения каждого из классов известны, то искомый алгоритм можно выписать в явном аналитическом виде. Более того, этот алгоритм оптимален, то есть обладает минимальной вероятностью ошибок. На практике плотности распределения классов, как правило, не известны. Их приходится оценивать (восстанавливать) по обучающей выборке. В результате байесовский алгоритм перестаёт быть оптимальным, так как восстановить плотность по выборке можно только с некоторой погрешностью. Чем короче выборка, тем выше шансы подогнать распределение под конкретные данные и столкнуться с эффектом переобучения. Байесовский подход к классификации является одним из старейших, но до сих пор сохраняет прочные позиции в теории распознавания. Он лежит в основе многих достаточно удачных алгоритмов классификации. К числу байесовских методов классификации относятся:

- квадратичный классификатор
- линейный дискриминант Фишера
- наивный байесовский классификатор
- метод парзеновского окна
- разделение смеси вероятностных распределений (EM-алгоритм)
- метод потенциальных функций или метод радиальных базисных функций

– метод ближайших соседей

- Нейронная сеть

  Нейронные сети выдают ответ вида: $y(\mathbf{x}, \mathbf{w}) = \mathbf{f}\left(\sum_{\mathbf{j=1}}^{\mathbf{N}} \mathbf{w_j}\phi_{\mathbf{j}}(\mathbf{x})\right)$, где f — нелинейная функция активации, $\mathbf{w}$ — вектор весов, ϕ — нелинейные базисные функции. Обучение нейронных сетей состоит в настройке весов а также базисных функций. Виды:

  – персептрон

  – многослойный персептрон

  – гибридная сеть встречного распространения

- Линейный разделитель

  Способ решения задач классификации, когда решение принимается на основании линейного оператора над входными данными. Класс задач, которые можно решать с помощью линейных классификаторов, обладают, соответственно, свойством линейной сепарабельности. Операцию линейной классификации для двух классов можно себе представить как отображение объектов в многомерном пространстве на гиперплоскость, в которой те объекты, которые попали по одну сторону разделяющей линии, относятся к первому классу ("да"), а объекты по другую сторону - ко второму классу ("нет")).

  – линейный дискриминант Фишера

  – наивный байесовский классификатор

  – однослойный персептрон

  – логистическая регрессия

  – машина опорных векторов

- Индукция правил: Логические алгоритмы классификации представляют собой композиции простых, легко интерпретируемых правил
  - решающее дерево
  - решающий список
  - решающий лес
  - тестовый алгоритм
  - алгоритм вычисления оценок
- Алгоритмическая композиция:
  - взвешенное голосование
  - бустинг
  - бэггинг
  - метод комитетов
  - смесь экспертов
- Сокращение размерности: Методы сокращения размерности путем выбора наиболее существенных (несущественных) признаков.
  - селекция признаков
  - метод главных компонент
  - метод независимых компонент
  - многомерное шкалирование
- Выбор модели: Выбор наиболее подходящего способа решения задачи.
  - минимизация эмпирического риска

– структурная минимизация риска

– минимум длины описания

– скользящий контроль

– извлечение признаков

– самоорганизация моделей

– случайный поиск с адаптацией

– генетический алгоритм

Выводы к главе.

В первой главе была поставлена задача классификации, а также описаны особенности решения задачи и варианты применения полученных результатов в реальном мире. Далее был приведен список методов решения, применимых в задачах классификации с описанием основных идей.

Глава 2. Задача «структура-свойство».

Постановка задачи «структура-свойство».

Термин хемоинформатика – применение методов информатики для решения химических проблем – был введён в употребление Ф. К. Брауном. Точнее, это совместное использование информационных ресурсов для преобразования данных в информацию и информации в знания для быстрейшего принятия наилучших решений при поиске соединений-лидеров в разработке лекарств и их оптимизации. Эта наука находится на пересечении химии и информатики. В основе хемоинформатики лежит представление о химическом пространстве — совокупности всех доступных химических объектов (химических соединений, реакций, смесей, растворов, каталитических систем, материалов и др.).

Отличительной особенностью хемоинформатики является то, что в её рамках прогнозирование свойств химических объектов осуществляется путём переноса (интерполяции) известных значений свойств от сходных химических объектов. В большинстве случаев химические объекты представимы в виде молекулярных графов, и поэтому методы теории графов находят широкое применение в хемоинформатике. Традиционный подход к обработке химической информации, однако, состоит в отображении химического пространства на дескрипторное пространство, образуемое вычисляемыми для каждого химического объекта векторами молекулярных дескрипторов — числовых характеристик, описывающих химические объекты (в особенности, молекулярные графы). Это дает возможность применять методы математической статистики и машинного обучения (в том числе, интеллектуального анализа

данных) для работы с химическими объектами. Сферы приложения хемоинформатики: прогноз физико-химических свойств химических соединений (в частности, липофильности, водорастворимости), свойств материалов, токсикологическая и биологическая активность, ADME/T, экотоксикологические свойства, разработка новых лекарственных препаратов и материалов.

Одной из основных задач хемоинформатики является поиск количественных соотношений структура-свойство[2] — процесс построения моделей, позволяющих по структурам химических соединений предсказывать их свойства. За моделями, позволяющими прогнозировать количественные характеристики биологической активности, исторически закрепилось англоязычное название Quantitative Structure-Activity Relationship (QSAR). При качественном описании соотношений между структурами химических соединений и их биологической активностью употребляют англоязычный термин Structure-Activity Relationship (SAR). Поиск количественных соотношений структура-свойство основан на применении методов математической статистики и машинного обучения для построения моделей, позволяющих по описанию структур химических соединений предсказывать их свойства (физические, химические, биологическую активность). При прогнозировании свойств на качественном уровне (например, будет ли данное химическое соединение обладать данным видом биологической активности) говорят о решении классификационной задачи, тогда как при прогнозировании числовых значений свойств говорят о решении регрессионной задачи. Описание структур химических соединений для этих целей может быть векторным либо невекторным (графовым).

При векторном описании химической структуре ставится в соот-

ветствие вектор молекулярных дескрипторов, каждый из которых представляет собой инвариант молекулярного графа. Моделирование свойств при невекторном описании химических соединений осуществляется либо при помощи нейронных сетей специальных архитектур, позволяющих работать непосредственно с матрицами смежности молекулярных графов, либо при помощи ядерных (kernel) методов с использованием специальных графовых (либо химических, фармакофорных) ядер.

Задача структура-свойство существенно распадается на две части: формирование МД-матрицы каким-либо способом и методы машинного обучения и анализа данных для улучшения качества выборки и для уточнения результатов.

Для решения первой поставленной задачи можно использовать векторные либо графовые способы описания химических соединений.

Для векторного способа наборы молекулярных дескрипторов можно разделить на следующие основные виды[2]:

- **Фрагментные дескрипторы** - существуют в двух основных вариантах — бинарном и целочисленном. Бинарные фрагментные дескрипторы показывают, содержится ли данный фрагмент (подструктура) в структурной формуле (то есть содержится ли данный подграф в молекулярном графе, описывающем данное химическое соединение), тогда как целочисленные фрагментные дескрипторы показывают, сколько раз данный фрагмент (подструктура) содержится в структурной формуле (то есть сколько раз содержится данный подграф в молекулярном графе, описывающем данное химическое соединение).

Уникальная роль фрагментных дескрипторов заключается в том, что они образуют базис дескрипторного пространства, то есть любой молекулярный дескриптор (и любое молекулярное свойство), являющийся инвариантом молекулярного графа, может быть однозначно разложен по этому базису.

- **Физико-химические дескрипторы** – это числовые характеристики, получаемые в результате моделирования физико-химических свойств химических соединений, либо величины, имеющие четкую физико-химическую интерпретацию. Наиболее часто используются в качестве дескрипторов: липофильность, молярная рефракция, молекулярный вес, дескрипторы водородной связи, молекулярные объёмы и площади поверхностей.

- **Квантово-химические дескрипторы** – это числовые величины, получаемые в результате квантово-химических расчетов. Наиболее часто в качестве дескрипторов используются: энергии граничных молекулярных орбиталей, частичные заряды на атомах и частичные порядки связей, индексы реакционной способности Фукуи (индекс свободной валентности, нуклеофильная и электрофильная суперделокализуемость), энергии катионной, анионной и радикальной локализации, дипольный и высшие мультипольные моменты распределения электростатического потенциала.

- **Дескрипторы молекулярных полей** – это числовые величины, аппроксимирующие значения молекулярных полей путём вычисления энергии взаимодействия пробного атома, помещенного в узел решетки, с текущей молекулой.

- **Константы заместителей** – впервые были введены Л. П. Гамметом в рамках уравнения, получившего его имя, которое связывает константы скорости реакции с константами равновесия для некоторых классов органических реакций. Константы заместителей вошли в практику QSAR после появления уравнения Ганча-Фуджиты, связывающего биологическую активность с константами заместителей и значением липофильности. В настоящее время известно несколько десятков констант заместителей.

- **Фармакофорные дескрипторы** – показывают, могут ли простейшие фармакофоры, состоящие из пар или троек фармакофорных центров со специфицированным расстоянием между ними, содержаться внутри анализируемой молекулы.

- **Дескрипторы молекулярного подобия** – указывают на меру сходства (молекулярного подобия) с соединениями из обучающей выборки.

- **Топологические индексы** – инварианты молекулярного графа в задачах компьютерной химии, некоторое числовое значение (или набор значений), характеризующее структуру молекулы. Обычно топологические индексы не отражают кратность химических связей и типы атомов, атомы водорода не учитываются. К наиболее известным топологическим индексам относятся индекс Хосои, индекс Винера, индекс Рандича, индекс Балабана и другие.

Включая и выключая дескрипторы можно наблюдать, как меняются свойства химических соединений. Также отметим, что при добавлении новых маркеров молекулы будут переставать быть рав-

ными, это будет подробнее рассмотрено в следующей главе.

Моделирование свойств при невекторном описании химических соединений происходит при помощи нейронных сетей специальных архитектур, позволяющих работать с матрицами смежности для молекулярных графов, либо при помощи ядерных методов с использованием специальных графовых ядер. Примерами служащих для этой цели нейронных сетей со специальной архитектурой являются BPZ, ChemNet, CCS, MolNet и другие. Примерами служащих для этой цели графовых ядер являются Marginalized graph kernel, Optimal assignment kernel, Pharmacophore kernel и другие.

Описание М-графа с помощью формирования дескрипторов.

Будем рассматривать молекулу как размеченный граф[2]. Его вершинами будут атомы, из которых состоит молекула, а ребрами — валентные связи между парами атомов[30]. Каждое ребро и каждая вершина имеют уникальное описание, однозначно кодирующие их характеристики, такие как координаты, заряд ядра или символ химического элемента для атомов и кратность или длина для связей. Построим обучающую выборку S, которая состоит из N объектов и представляет собой список пар: $TS = (X_1, C_1), ..., (X_N, C_N)$, где i-ый объект представлен в виде вектора $X_i = (X_{i1}, X_{i2}, ..., X_{i3})$, а C_i – внешний признак, задающий принадлежность i-го объекта к одному из классов. Также необходимо найти функцию , которая по молекулярному графу наилучшим образом определяла бы для него значение исследуемого свойства. Требуется построить «функцию качества» F такую, что

на обучающей выборке ее значения были бы максимально близки к реальной внешней классификации C, заданной учителем. Данная формулировка определяет задачу «структура-свойство» или QSAR-задачу (Quantitative Structure Activity Relationships).

Отметим, что у нее есть особенности, связанные с химической интерпретацией[8]:

- C – внешний признак для выборки – может быть числом или промежутком на действительной прямой, а также классификатором для какого-либо свойства химического соединения.
- Классифицирующая функция F обычно имеет линейный вид.
- Признаками-столбцами, по которым определяются свойства химического соединения, являются простые графы, составленные особым способом.

Для того, чтобы найти оптимальный вид функции F, мы, очевидно, можем работать именно с ней, и за счет усложнения ее вида достигнуть адекватных результатов. Но более рациональным способом является выделение в М-графе дескрипторов с постепенным увеличением детализации. Другими словами, мы будем использовать не фиксированный вектор признаков, а создавать алфавит дескрипторов на основе анализа соединения и далее учитывать наличие в М-графе пар, троек, четверок фрагментов из заданного алфавита элементов описания.Тем самым, мы приходим к дополнительной подзадаче: формирование пространства признаков.

Рассмотрим набор фрагметов без циклов М-графов $H = h_i, i = 1..L$. **Структурным дескриптором** $G(h_i)$ будем называть пару (h_i, X_i), где X_i – это количество повторений фрагмента в рассматриевом графе G. Строку $(X_1, X_2, ..., X_L)$ назовем **структурным**

спектром $SS(G|H)$.

Фрагменты будем кодировать строкой, составленной из меток атомов, фходящих в данный фрагмент (записаны в лексикографическом порядке). Таким образом, имена фрагментов будут совпадать тогда и только тогда, когда сами фрагменты изоморфны. Также необходимо изначально построить библиотеки из всех фрагментов длины k и на их основании строить структурный спектр графа G.

Формирование матрицы „Молекула-Дескриптор“.

Для начала работы необходимо разметить каждый М-граф, то есть присвоить каждой его вершине соответствующую метку. Удобным способом маркирования является следующий. Для каждого атома определяем три маркера:

- **p** – степень атома, от одого до шести.
- **b** – тип химической связи атома, s – все связи одинарные, d – есть двойная связь, t – есть тройная связь, w – две двойные связи.
- **r** – положенине атома в кольцевой системе, c – цепной, r – только кольцевой, s – кольцевой с заместителем.

Метка атома будет иметь вид $NNpbr$, где NN – это имя атома.

Далее, для каждого М-гафа строим список имен его фрагментов и формируем k-спектры для каждого графа. Объединяем полученные списки: упорядочиваем и убираем лишние имена, встречающиес в нескольких М-графах, после чего строим общий структурный спектр для получившегося набора имен.

Так получаем МД-матрицу, где каждая строка – описание одного графа с помощью всевозможных дескрипторов.

Анализ способа решения задачи.

В задачах машинного обучения[17] качество моделей[16] очень сильно зависит от данных. Но сами данные в реальных задачах редко бывают идеальными. Как правило, самих данных немного, количество доступных для анализа параметров ограничено, в данных шумы и пропуски. Выше обсуждалось то, как удобнее всего собирать данные для решения нашей задачи. Теперь обсудим что делать перед непосредственным построением регрессии или решением задачи классификации[4].

Стоит начать с проверки взаимных корреляций. Но некоторые признаки могут просто линейно выражаться друг через друга. В этом случае также нужно оставить только один. В случае прямой линейной зависимости смысла оставлять оба параметра из коррелирующей пары нет. Отмечу, что просто к функциональной зависимости это не относится. И, наконец, посчитаем features importance. Делать это нужно по двум причинам: во-первых, откровенно слабые информационные признаки могут непродуктивно нагружать наши вычислительные ресурсы, не привнося полезной информации, во-вторых, нам нужно найти самые важные признаки и проанализировать их. О том, какие есть способы отбора таких признаков, будем рассказано в следующей главе.

Далее следует задуматься о том, что наша задача состоит в исследовании результатов при поиске кластеров, то есть наборов таких объектов, которые обладают некоторой общностью согласно своему описанию. Для количественной оценки этой общности вво-

дят расстояние между объектами в заданной метрике. Таким образом, кластерный анализ обучающей выборки состоит в разбиении объектов на непересекающиеся наборы, называемые кластерами, так, чтобы кластер состоял из «близких» в данной метрике объектов, а расстояния между кластерами существенно отличались. В машинном обучении задача кластерного анализа относится к задачам «обучения без учителя». Поэтому одна из ключевых задач узнавания, или распознавания состоит в поиске такой метрики, то есть такого «правильного» набора признаков, описывающих исследуемые объекты, в котором объекты одного класса «близки» друг к другу. В этом случае мы получаем «дерево решений» как форму задания классификатора. Для нового объекта определяется кластер, а затем принимается решение о классификации нового объекта на основе его принадлежности кластеру.

Описанный выше подход может быть естественно обобщен на случай поиска регрессии. Для правильной интерпретации весовых коэффициентов регрессии требуется нормальное распределение признаков, на которых строится регрессия. Поэтому, если обучающая выборка имеет кластерную структуру, то в каждом найденном кластере следует строить собственную регрессию.

Таким образом, приходим к понятию «Обобщенного дерева решений» (ОДР). В вершине ОДР находится классификатор, определяющий, к какому кластеру принадлежит новый объект. Далее используется регрессия, построенная для данного кластера. Здесь возникает ситуация, когда возможен «отказ от прогнозирования» – если объект не принадлежит ни одному кластеру, то есть значительно удален от всех кластеров.

Конкретные алгоритмы построения кластеров и отбора главных

элементов мы рассмотрим в следующей главе. А далее применим их на выборке циклооксигеназа-2 (ЦОГ-2) – это ген, кодирующий действие циклооксигеназ (ЦОГ) – группа ферментов, участвующих в синтезе простаноидов. Фармакологическое ингибирование ЦОГ ослабляет симптомы воспаления и боли, примерами таких ингибиторов являются аспирин и ибупрофен. Предсказывать будем свойство IC50 – это концентрация полумаксимального ингибирования – показатель эффективности нестероидных противовоспалительных препаратов при ингибировании изоформ ЦОГ.

Выводы к главе.

В данной главе была поставлена задача поиска количественных отношений структура-свойство. Были описаны основные области изучения в хемоинформатике, а также были описаны способы обработки данных, которые, как мы предполагаем, должны улучшить результаты. Описаны два метода учета аргументов для выбора дескрипторов, вносящих наибольший вклад, несколько способов кластеризации, а также рассмотрен способ понижения размерности для того, чтобы иметь возможность корректно графически изображать многомерные данные.

Глава 3. Описание метода опорнвых векторов.

Далее в работе будет рассмотрен метод опорных векторов (англ. SVM, Support Vector Machine) для задачи классификации, описанной ранее. Будет описана основная идея алгоритма, произведена настройка его весов и разобрана реализация метода своими руками. Рассмотрим плюсы и минусы алгоритма, его модификации с помощью различных ядерных функций.

Постановка задачи.

Будем решать задачу бинарной (когда класса всего два) классификации.Разделим выборку на два множества: обучающую и тестовую. Сначала алгоритм выполняется на объектах из обучающей выборки, для которых заранее известны значения классов. Далее уже обученный алгоритм предсказывает метку класса для каждого объекта из тестовой выборки. Значения меток классов являются множеством $Y = +1; -1$. элемент выборки — вектор с n признаками $x = (x_1, ..., x_n)$ в пространстве R^n [6]. При обучении алгоритм строит функцию $F(x) = y$, которой на вход подается аргумент x — объект из пространства R^n и возвращает метку класса y.

Алгоритм.

Суть SVM как классификатора — поиск уравнения разделяющей гиперплоскости $w_1x_1 + w_2x_2 + ... + w_nx_n + w_0 = 0$ в пространстве R^n, которая бы разделила два класса неким оптимальным образом. Общий вид преобразования F объекта x в метку класса Y выглядит как $F(x) = sign(< wx > -w_0)$. Где $x = (x_1, ..., x_n)$ —

признаковое описание объекта x, вектор $w = (w_1, ..., w_n) \in R^n$ и скалярный порог $w_0 \in R$ являются параметрами алгоритма. Уравнение $<w, x> = w_0$ описывает гиперплоскость, разделяющую классы в пространстве R^n. Назовем его пороговой функцией. После настройки весов алгоритма w и b (обучения), все объекты, попадающие по одну сторону от построенной гиперплоскости, будут предсказываться как первый класс, а объекты, попадающие по другую сторону — второй класс.

Очевидно, что разделяющую гиперплоскость можно строить разными способами, но в SVM веса w и b подбираются так, чтобы объекты классов с каждой стороны лежали как можно дальше от разделяющей гиперплоскости. Другими словами, алгоритм максимизирует зазор (англ. margin) между гиперплоскостью и объектами классов, которые расположены ближе всего к ней. Такие объекты и называют опорными векторами. Отсюда и название алгоритма.

Линейно разделимая выборка.

Сначала рассмотрим случай полностью разделимой выборки, т.е. случай, когда обучение возможно провести без ошибок. Решим задачу нахождения наиболее удачного разделения множества векторов - элементов выборки - на два класса с помощью линейной решающей функции. Обозначим (Ξ, Y), где $\Xi = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ — обучающую выборку, а $Y = (y_1, ..., y_N)$ — множество меток двух классов ω_1 и ω_2. Хотим по обучающей выборке построить линейную решающую функцию, т.е. найти такую линейную функцию $f(\mathbf{x})$, которая удовлетворяла бы условию $f(\mathbf{x}_i) > 0$ для всех $\mathbf{x}_i \in \omega_1$,

$f(\mathbf{x}_i) < 0$ для всех $\mathbf{x}_i \in \omega_2$.

Исходя из того, что метками считаем значения $+1; -1$, $f(\mathbf{x})$, будет удовлетворять условию (1):

$$y_i f(\mathbf{x}_i) > 0 \qquad \forall \mathbf{x}_i \in \Xi$$

Линейным преобразованием, если необходимо, можем привести систему к виду:

$$y_i f(\mathbf{x}_i) > 1 \qquad \forall \mathbf{x}_i \in \Xi$$

.

Кроме того, так как $f(\mathbf{x})$ — линейная функция, то последняя система неравенств примет вид (2):

$$y_i((\mathbf{w}, \mathbf{x}_i) + b) \geq 1, \quad i = 1, ..., N$$

, где $\mathbf{w}$ — вектор весовых коэффициентов, b — некоторое число. Тогда разделяющей два класса гиперплоскостью будет $(\mathbf{w}, \mathbf{x}) + b = 0$.

Нетрудно видеть, что и все гиперплоскости вида $(\mathbf{w}, \mathbf{x}) + b' = 0$, где $b' \in (b-1, b+1)$, также будут разделяющими. Расстояние между граничными гиперплоскостями $(\mathbf{w}, \mathbf{x}) + b - 1 = 0$ и $(\mathbf{w}, \mathbf{x}) + b + 1 = 0$ равно: $\frac{2}{\|\mathbf{w}\|}$. Действительно, $\left(\frac{\mathbf{w}}{\|\mathbf{w}\|}, \mathbf{x}\right) + \frac{b-1}{\|\mathbf{w}\|} = 0$ и $\left(\frac{\mathbf{w}}{\|\mathbf{w}\|}, \mathbf{x}\right) + \frac{b+1}{\|\mathbf{w}\|} = 0$ — нормальные уравнения этих гиперплоскостей. Тогда $p_1 = \frac{b-1}{\|\mathbf{w}\|}$ и $p_2 = \frac{b+1}{\|\mathbf{w}\|}$ — расстояния от этих гиперплоскостей до начала координат и $\frac{2}{\|\mathbf{w}\|}$ — расстояние между гиперплоскостями. На самих граничных плоскостях может находиться некоторое число опорных векторов.

Для полного разделения классов хотим, чтобы расстояние между разделяющими гиперплоскостями было максимально возмож-

ным, т.е. норма весовых коэффициентов (в знаменателе) $\|\mathbf{w}\|$ была как можно меньше. Таким образом, ставится задача нахождения минимума квадратичного функционала $0.5(\mathbf{w},\mathbf{w})$ (коэффициент 0.5 введем для удобства дифференцирования в дальнейшем) в выпуклом многограннике, задаваемым системой неравенств (2). В выпуклом множестве квадратичный функционал всегда имеет единственный минимум (если это множество не пусто). Из теоремы Куна — Таккера[16] следует, что решение этой оптимизационной задачи равносильно поиску седловой точки лагранжиана

$$L(\mathbf{w}, b, \lambda) = 0.5(\mathbf{w}, \mathbf{w}) - \sum_{i=1}^{N} \lambda_i (y_i((\mathbf{w}, \mathbf{x}_i) + b) - 1) \rightarrow \min_{\mathbf{w},b} \max_{\lambda}$$

в ортанте по множителям Лагранжа $\lambda_i \geq 0 \quad (i = 1, ..., N)$, при условии, что

$$\lambda_i \left(y_i((\mathbf{w}, \mathbf{x}_i) + b) - 1\right) = 0, \quad i = 1, ..., N$$

. Последнее условие равносильно тому, что (3):

$$\lambda_i = 0$$

или

$$y_i((\mathbf{w}, \mathbf{x}_i) + b) - 1 = 0, \quad i = 1, ..., N$$

Из необходимых условий существования седловой точки (полагая $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{in})$) имеем:

$$\Gamma \quad \begin{cases} 0 = \frac{\partial L}{\partial w_j} = w_j - \sum\limits_{i=1}^{N} \lambda_i y_i x_{ij}, \quad j = 1, ..., n, \\ 0 = \frac{\partial L}{\partial b} = \sum\limits_{i=1}^{N} \lambda_i y_i. \end{cases}$$

Откуда следует, что вектор $\mathbf{w}$ следует искать в виде (4):

$$\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$$

, причем (5):

$$\sum_{i=1}^{N} \lambda_i y_i = 0$$

.

В силу (3) в сумму (4) с ненулевыми коэффициентами λ_i входят только те векторы, для которых $y_i((\mathbf{w}, \mathbf{x}_i) + b) - 1 = 0$. Такие векторы называют опорными, так как это именно те векторы, через которые будут проходить граничные гиперплоскости, разделяющие классы. Для найденного весового вектора $\mathbf{w}$ смещение b можно вычислить как $b = {y_s}^{-1} - (\mathbf{w}, \mathbf{x}_s)$ для любого опорного вектора $\mathbf{x}_s$.

Найдем значения множителей Лагранжа, как критических точек лагранжиана. Для этого подставим (4) и (5) в лагранжиан, получим

$$L(\mathbf{w}, b, \lambda) = 0.5(\mathbf{w}, \mathbf{w}) - \sum_{i=1}^{N} \lambda_i (y_i((\mathbf{w}, \mathbf{x}_i) + b) - 1) =$$

$$= 0.5(\mathbf{w}, \mathbf{w}) - \left((\mathbf{w}, \mathbf{w}) - \sum_{i=1}^{N} \lambda_i \right) = \sum_{i=1}^{N} \lambda_i - 0.5(\mathbf{w}, \mathbf{w}) =$$

$$= \sum_{i=1}^{N} \lambda_i - 0.5 \sum_{i,j=1}^{N} \lambda_i \lambda_j y_i y_j (\mathbf{x}_i, \mathbf{x}_j) = \sum_{i=1}^{N} \lambda_i - 0.5 \left\| \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i \right\|^2$$

.

Таким образом, задача сводится к нахождению критических точек функции (6):

$$\Phi(\lambda) = \sum_{i=1}^{N} \lambda_i - 0.5 \left\| \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i \right\|^2$$

. Так как эта функция представляет собой разность линейной и квадратичной функций, причем квадратичная функция отрицательно определена, то требуется найти наибольшее значение функции $\Phi(\lambda)$ при условии $\sum_{i=1}^{N} \lambda_i y_i = 0$ в области $\lambda_i \geq 0 \;\; (i = 1, ..., N)$.

Линейно неразделимая выборка.

Для того, чтобы алгоритм мог работать в случае, если классы линейно неразделимы, позволим ему допускать ошибки на обучающей выборке. Введём набор дополнительных переменных $\xi_i \geq 0$, характеризующих величину ошибки на объектах $\mathbf{x}_i, \quad 1 \leq i \leq n$. Возьмём за отправной точкой задачу нахождения минимума квадратичного функционала, аналогичную предыдущей главе, введём в минимизируемый функционал штраф за суммарную ошибку:

$$\begin{cases} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i \to \min_{w,b,\xi_i} \\ c_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1 - \xi_i, \quad 1 \leq i \leq n \\ \xi_i \geq 0, \quad 1 \leq i \leq n \end{cases}$$

Коэффициент C здесь — это параметр метода, регулирующий отношение максимизации ширины разделяющей полосы к минимизации суммарной ошибки.

Аналогично прошлой главе, по теореме Куна-Таккера сводим задачу к поиску седловой точки функции Лагранжа:

$$\begin{cases} \mathbf{L}(\mathbf{w}, \mathbf{b}, \xi; \lambda, \eta) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \lambda_\mathbf{i}(c_i((\mathbf{w} \cdot \mathbf{x_i}) - b) - 1) - \\ - \sum_{i=1}^{n} \xi_\mathbf{i}(\lambda_\mathbf{i} + \eta_\mathbf{i} - C) \to \min_{w,b,\xi} \max_{\lambda,\eta} \\ \xi_\mathbf{i} \geq 0, \lambda_\mathbf{i} \geq 0, \eta_\mathbf{i} \geq 0, \quad 1 \leq i \leq n \\ \left[\begin{array}{l} \lambda_\mathbf{i} = 0 \\ c_i(\mathbf{w} \cdot \mathbf{x_i} - b) = 1 - \xi_i, \end{array} \right. \quad 1 \leq i \leq n \\ \left[\begin{array}{l} \eta_\mathbf{i} = 0 \\ \xi_\mathbf{i} = 0, \end{array} \right. \quad 1 \leq i \leq n \end{cases}$$

Сведём задачу к эквивалентной:

$$\begin{cases} -\mathbf{L}(\lambda) = -\sum_{i=1}^{n} \lambda_\mathbf{i} + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_\mathbf{i} \lambda_\mathbf{j} c_i c_j (\mathbf{x_i} \cdot \mathbf{x_j}) \to \min_\lambda \\ 0 \leq \lambda_\mathbf{i} \leq \mathbf{C}, \quad 1 \leq i \leq n \\ \sum_{i=1}^{n} \lambda_\mathbf{i} c_i = 0 \end{cases}$$

В решении практических задач применяется именно метод для неразделимой выборки, так как в общем случае данные обычно лиейно неразделимы. Этот вариант алгоритма называют алгоритмом с мягким зазором (soft-margin SVM), тогда как для линейно разделимого случая имеет место жёсткий зазор (hard-margin SVM).

Теперь ненулевыми $\lambda_\mathbf{i}$ будут обладать не только опорные вектора, но и вектора со штрафами. Это наталкивает на мысль об удалении из выборки шумовых выбросов, так как в случае их наличия построенный алгоритм будет опираться на шум.

Константу C можно искать вручную либо по критерию скользящего контроля. Но это долгий процесс, так как задачу приходится решать заново при каждом значении C, поэтому в случае хороших результатов но случайном C можно остановиться на выбранном

значении.

Ядра и спрямляющие пространства.

Существует ещё один подход к решению проблемы линейной неразделимости. Это переход от исходного пространства признаковых описаний объектов X к новому пространству H с помощью некоторого преобразования $\psi : X \to H$. Если пространство H имеет достаточно высокую размерность, то можно надеяться, что в нём выборка окажется линейно разделимой[7]. Пространство H называют спрямляющим.

Если предположить, что признаковыми описаниями объектов являются векторы $\psi(x_i)$, а не векторы x_i, то построение SVM проводится точно так же, как и ранее. Единственное отличие состоит в том, что скалярное произведение $\langle \mathbf{x}, \mathbf{x}' \rangle$ в пространстве X всюду заменяется скалярным произведением $\langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$ в пространстве H. Отсюда вытекает естественное требование: пространство H должно быть наделено скалярным произведением, в частности, подойдёт любое евклидово, а в общем случае и гильбертово, пространство.

Отсюда можем перейти к определению функции ядра. Функция $K : X \times X \to R$ называется ядром (kernel function), если она представима в виде $K(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$ при некотором отображении $\psi : X \to H$, где H — пространство со скалярным произведением. Постановка задачи, и сам алгоритм классификации зависят только от скалярных произведений объектов, но не от самих признаковых описаний. Это означает, что скалярное произведение $(\mathbf{x}, \mathbf{x}')$ можно формально заменить ядром $K(\mathbf{x}, \mathbf{x}')$. Более

того, можно вообще не строить спрямляющее пространство H в явном виде, и вместо подбора отображения ψ заниматься непосредственно подбором ядра.

Любая ли функция двух аргументов $K(\mathbf{x}, \mathbf{x}')$ может исполнять роль ядра? Теорема Мерсера дает ответ на этот вопрос и показывает, что класс допустимых ядер достаточно широк.

Теорема: Функция $K(\mathbf{x}, \mathbf{x}')$ является ядром тогда и только тогда, когда она симметрична, $K(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}', \mathbf{x})$, и неотрицательно определена. Функция $K(\mathbf{x}, \mathbf{x}')$ неотрицательно определена, если для любой конечной выборки $X_p = (x_1, ..., x_p)$ из X матрица $K = K(x_i, x_j)$ размера $p \times p$ неотрицательно определена: $z^T K z \geq 0 \forall z \in \mathbf{R}^p$.

Проверка неотрицательной определённости функции в практических ситуациях может оказаться делом нетривиальным. Часто ограничиваются перебором конечного числа функций, про которые известно, что они являются ядрами. Среди них выбирается лучшая, но очевидно, что это не оптимальное решение. На сегодняшний день проблема выбора ядра, оптимального для данной конкретной задачи, остаётся открытой.

Выводы к главе.

В данной главе была поставлена задача, решаемая с помощью представленного метода опорных векторов. Также были описаны частный (для линейно разделимой выборки) и общий (для линейно неразделимой выборки) методы решения задачи. Был предложен альтернативный способ решения в общем случае с помощью различных ядерных функций.

Глава 4. Реализация метода и анализ полученных результатов.

Реализация метода опорных векторов.

Для реализации метода на языке Python будем использовать библиотеки pandas, numpy, mathplotlib, sklearn и scipy. Далее опишем вспомогательные функции.

Для подсчета точности используем формулу:

$$1 - \frac{\sum\limits_{i \in (actual \setminus predicted)} x_i}{\sum\limits_{x \in actual} 1}$$

. Также реализуем класс, вычисляющий ядерную функцию по выбранной формуле, и сам метод, пользуясь рассуждениями в предыдущих главах, а именно максимизируя лагранжиан при описанных условиях.

Далее загрузим данные и разделим их случайным образом на тестовую и трернировочную выборки. Для работы написанной модели необходимо выбрать параметры: ядро (изначально работаем с rbf), γ — это ширина rbf ядра, C — параметр регуляризации (он в том числе контролирует соотношение между гладкой границей и корректной классификацией рассматриваемых точек). Запустим метод в цикле, меняя γ как [0.1, 0.5, 1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100] и C как [1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]:. Таким образом, получим результаты 154 запусков, они представлены в Приложении 1. Выберем наиболее удачные из них:

```
0.1    10     Classification accuracy:  0.7446808510638299
```

Далее имеет смысл удалить из выборки то, что ранее было названо шумами. Запустим на "чистых"данных метод с выбранными параметрами $\gamma = 0.1$ и C=10 порядка 20 раз, чтобы сравнить результаты на разных запусках с одними и теми же параметрами. Среднее значение для точности 0.8572289156626507. Подробнее полученные значения можно изучить в Приложении 2.

Сравнение значений при использовании разных ядерных функций.

Перейдем к сравнению результатов, полученных с использованием разных ядерных функций.

Перечислим примеры функций, которые могут называться ядрами:

- линейное ядро $k(x, y) = x^T y + c$
- полиномиальное ядро $k(x, y) = (\alpha x^T y + c)^d$
- ядро Гаусса k(x, y) = $exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$
- экспонециальное ядро $k(x, y) = exp\left(-\frac{\|x-y\|}{2\sigma^2}\right)$
- ядро Лапласа $k(x, y) = exp\left(-\frac{\|x-y\|}{\sigma}\right)$
- ядро ANOVA $k(x, y) = \sum_{k=1}^{n} exp(-\sigma(x^k - y^k)^2)^d$
- сигмоида $k(x, y) = \tanh(\alpha x^T y + c)$
- радиальное квадратичное ядро $k(x, y) = 1 - \frac{\|x-y\|^2}{\|x-y\|^2+c}$
- мультиквадратное ядро $k(x, y) = \sqrt{\|x - y\|^2 + c^2}$
- обратное мультиквадратное ядро $k(x, y) = \frac{1}{\sqrt{\|x-y\|^2+\theta^2}}$

Запустим на каждом и увидим, что точность выше на ядре, являющимся функцией Гаусса (например на линейном ядре точность

0.7027027027027026, а на полиномиальном 0.8108108108108107).То есть наиболее точно форма классов в нашей выборке описывается радиальной базисной функцией.

В приложении 3 приведены сравнительные результаты запуска на "чистой"выборке для rbf-функции.

Также сравним результаты со стандартным методом библиотеки на том же разбиении: 0.8313253012048193. Можем считать, что результаты близки к истинным.

Выводы к главе. Анализ полученных результатов.

В данной главе рассматривается работа алгоритма с реальными данными. Методом перебора подбираются подходящие параметры для работы алгоритма. Также проводятся эксперименты с ядерными функциями и сравнение результатов с встроенным методом.

Заключение.

Основной задачей, которая решена в данной курсовой работе, является иследование выборки циклооксигеназа-2 (ЦОГ-2) и предсказывание концентрации полумаксимального ингибирования. Для решения этой задачи был предложен способ формирования МД-матрицы на основе М-графа каждого соединения. Это позволило однозначно закодировать каждый М-граф для дальнейшего анализа.

Далее был описан метод опорных векторов. Для его работы принципиален момент линейной разделимости выборки, который рассмотрен отдельно. Если выборка изначально линейно неразделима, то задача сводится к поиску минимума квадратичного функционала либо напрямую через добавление условий на введенные параметры отношения ширины разделяющей полосы к минимизации суммарной ошибки, либо через спрямление пространства с помощью ядерной функции. На основании полученных результатов можно сделать вывод о плюсах метода:

- хорошо работает с пространством признаков большого размера;
- хорошо работает с данными небольшого объема;
- алгоритм максимизирует разделяющую полосу, которая, как подушка безопасности, позволяет уменьшить количество ошибок классификации;
- так как алгоритм сводится к решению задачи квадратичного программирования в выпуклой области, то такая задача всегда имеет единственное решение (разделяющая гиперплос-

кость с определенными гиперпараметрами алгоритма всегда одна).

И минусах метода:

- долгое время обучения (для больших наборов данных);
- неустойчивость к шуму: выбросы в обучающих данных становятся опорными объектами-нарушителями и напрямую влияют на построение разделяющей гиперплоскости;
- не описаны общие методы построения ядер и спрямляющих пространств, наиболее подходящих для конкретной задачи в случае линейной неразделимости классов.

В дальнейшем, планируется продолжение работы по данной теме и усовершенстование применяемого метода путем доведения метода опорных векторов до следующих возможных модификаций:

- Метод релевантных векторов (Relevance Vector Machine, RVM)
- Переход к l-норме (1-norm SVM, LASSO SVM)
- Doubly Regularized SVM (ElasticNet SVM)
- Support Features Machine (SFM)
- Relevance Features Machine (RFM)

Библиография.

1) Кумсков М.И. Выбор формальных элементов, покрывающий кластер молекулярных графов в метрике, заданной на их структурных символьных спектрах. // В сб. «Модели и архитектуры нейронных сетей в задаче «структура-свойство». М.: Изд-во «МАКСПРЕСС», 2020.

2) Кумсков М.И. «Методология Прогнозирования Свойств Химических Соединений И Ее Программная Реализация» - Автореферат докторской диссертации, М.: Вычислительный Центр РАН. - http://istina.msu.ru/dissertations/3541505/

3) Кумсков М.И., Митюшев Д.Ф. Применение метода группового учета аргументов (МГУА) для построения коллективных оценок свойств органических соединений на основе индуктивного перечисления их структурных спектров / Проблемы управления и информатики, 1996, No4.

4) Айвазян С. А., Енюков И. С., Мешалкин Л. Д. Прикладная статистика: основы моделирования и первичная обработка данных. — М.: Финансы и статистика, 1983

5) Айвазян С. А., Бухштабер В. М., Енюков И. С., Мешалкин Л. Д. Прикладная статистика: классификация и снижение размерности. — М.: Финансы и статистика, 1989.

6) Владимир Вьюгин. Математические основы теории машинного обучения и прогнозирования. — МЦМНО, 2013.

7) Aronszajn N. Theory of reproducing kernels // Transactions of the American Mathematical Society. 1950. V. 68. P. 337Ц404.

8) M. Hutter and J. Poland. Adaptive online prediction by following the perturbed leader // Journal of Machine Learning Research, 6:639–660, 2005.

9) Дидэ Э. Методы анализа данных. Подход, основанный на методе динамических сгущений — Пер. с англ., М.: Финансы и статистика, 1985.

10) Загоруйко Н.Г., Елкина В.Н., Лбов Г.С. Алгоритмы обнаружения

11) Себер Дж. Линейный регрессионный анализ. М.: Мир, 1980.

12) Журавлев Ю.И., Гуревич И.Б. Распознавание образов и распознавание изображений. / В сб. Распознавание. Классификация. Прогноз. Математические методы и их применение. Вып.2, М.: Наука, 1989

13) Харари Ф. Теория графов. - Пер. с англ., - М.: Мир, 1973.

14) Джулли А, Пал С. «Библиотека Keras – инструмент глубокого обучения. Реализация нейронных сетей с помощью библиотек Theano и TensorFlow» Пер. с англ., М:. ДМК Пресс, 2018.

15) Горелик А.Л., Гуревич И.Б., Скрипкин В.А. Современное состояние проблемы распознавания. Некоторые аспекты. - М.: Радио и связь, 1985.

16) Журавлев Ю.И. Об алгебраическом подходе к решению задач распознавания или классификации / Проблемы кибернетики, М.: Наука,1978.

17) Дэйвисон М. Многомерное шкалирование: Методы наглядного

представления данных. — Пер. с англ., М.: Финансы и статистика, 1988.

18) Айвазян С. А., Бухштабер В. М., Енюков И. С., Мешалкин Л. Д. Прикладная статистика. Классификация и снижение размерности.— М.: Финансы и статистика, 1989

19) Мейндональд Дж. Вычислительные алгоритмы в прикладной статистике. – Пер.с англ., М.: Финансы и статистика, 1988.

20) Бонгард М. М. Проблема узнавания. — М.: Физматгиз, 1967. 11 . Себер Дж. Линейный регрессионный анализ. — Пер. с англ., - М.: Мир, 1980.

21) Журавлев Ю.И., Гуревич И.Б. Распознавание образов и распознавание изображений. / В сб. Распознавание. Классификация. Прогноз. Математические методы и их применение. Вып.2, М.: Наука, 1989

22) Wei B. Q., Weaver L. H., Ferrari A. M., Matthews B. W., Shoichet B. K. Testing a flexible-receptor docking algorithm in a model binding site (англ.) // Journal of Molecular Biology, 2004.Vol. 337, no. 5).

23) Shestov A.M., Koptsova A.V., Kumskov M.I. The new molecular surface descriptors –(Jan.2011)

24) Хайкин, Саймон. Нейронные сети: полный курс, 2-е изд.: Пер. с англ. - М.: ООО "И.Д. Вильямс 2006.

25) Кристофидес Н. Теория графов. Алгоритмический подход - Пер. с англ. - М.: Мир, 1978.

26) Хардле В. Прикладная непараметрическая регрессия. - М.: Мир, 1993.

27) Melanie Mitchell. An Introduction to Genetic Algorithms. MIT Press, 1998.

Приложения.

Приложение 1:

```
gamma: C:
0.1    1      Classification accuracy:  0.648936170212766
0.1    10     Classification accuracy:  0.7446808510638299
0.1    20     Classification accuracy:  0.7340425531914894
0.1    30     Classification accuracy:  0.7021276595744681
0.1    40     Classification accuracy:  0.7021276595744681
0.1    50     Classification accuracy:  0.7340425531914894
0.1    60     Classification accuracy:  0.7340425531914894
0.1    70     Classification accuracy:  0.7340425531914894
0.1    80     Classification accuracy:  0.7446808510638299
0.1    90     Classification accuracy:  0.6914893617021276
0.1    100    Classification accuracy:  0.7340425531914894
0.5    1      Classification accuracy:  0.7127659574468085
0.5    10     Classification accuracy:  0.7021276595744681
0.5    20     Classification accuracy:  0.7021276595744681
0.5    30     Classification accuracy:  0.7021276595744681
0.5    40     Classification accuracy:  0.7021276595744681
0.5    50     Classification accuracy:  0.7021276595744681
0.5    60     Classification accuracy:  0.7021276595744681
0.5    70     Classification accuracy:  0.7021276595744681
0.5    80     Classification accuracy:  0.7021276595744681
0.5    90     Classification accuracy:  0.7021276595744681
0.5    100    Classification accuracy:  0.7021276595744681
1      1      Classification accuracy:  0.7234042553191489
1      10     Classification accuracy:  0.6595744680851063
```

```
1       20      Classification accuracy:  0.6595744680851063
1       30      Classification accuracy:  0.6595744680851063
1       40      Classification accuracy:  0.6595744680851063
1       50      Classification accuracy:  0.6595744680851063
1       60      Classification accuracy:  0.6595744680851063
1       70      Classification accuracy:  0.6595744680851063
1       80      Classification accuracy:  0.6595744680851063
1       90      Classification accuracy:  0.6595744680851063
1       100     Classification accuracy:  0.6595744680851063
5       1       Classification accuracy:  0.6914893617021276
5       10      Classification accuracy:  0.6595744680851063
5       20      Classification accuracy:  0.6595744680851063
5       30      Classification accuracy:  0.6595744680851063
5       40      Classification accuracy:  0.6595744680851063
5       50      Classification accuracy:  0.6595744680851063
5       60      Classification accuracy:  0.6595744680851063
5       70      Classification accuracy:  0.6595744680851063
5       80      Classification accuracy:  0.6595744680851063
5       90      Classification accuracy:  0.6595744680851063
5       100     Classification accuracy:  0.6595744680851063
10      1       Classification accuracy:  0.6595744680851063
10      10      Classification accuracy:  0.6595744680851063
10      20      Classification accuracy:  0.6595744680851063
10      30      Classification accuracy:  0.6595744680851063
10      40      Classification accuracy:  0.6595744680851063
10      50      Classification accuracy:  0.6595744680851063
10      60      Classification accuracy:  0.6595744680851063
10      70      Classification accuracy:  0.6595744680851063
```

```
10      80      Classification accuracy:  0.6595744680851063
10      90      Classification accuracy:  0.6595744680851063
10      100     Classification accuracy:  0.6595744680851063
20      1       Classification accuracy:  0.6595744680851063
20      10      Classification accuracy:  0.6595744680851063
20      20      Classification accuracy:  0.6595744680851063
20      30      Classification accuracy:  0.6595744680851063
20      40      Classification accuracy:  0.6595744680851063
20      50      Classification accuracy:  0.6595744680851063
20      60      Classification accuracy:  0.6595744680851063
20      70      Classification accuracy:  0.6595744680851063
20      80      Classification accuracy:  0.6595744680851063
20      90      Classification accuracy:  0.6595744680851063
20      100     Classification accuracy:  0.6595744680851063
30      1       Classification accuracy:  0.7021276595744681
30      10      Classification accuracy:  0.6595744680851063
30      20      Classification accuracy:  0.6595744680851063
30      30      Classification accuracy:  0.6595744680851063
30      40      Classification accuracy:  0.6595744680851063
30      50      Classification accuracy:  0.6595744680851063
30      60      Classification accuracy:  0.6595744680851063
30      70      Classification accuracy:  0.6595744680851063
30      80      Classification accuracy:  0.6595744680851063
30      90      Classification accuracy:  0.6595744680851063
30      100     Classification accuracy:  0.6595744680851063
40      1       Classification accuracy:  0.6595744680851063
40      10      Classification accuracy:  0.6595744680851063
40      20      Classification accuracy:  0.6595744680851063
```

```
40      30      Classification accuracy:  0.6595744680851063
40      40      Classification accuracy:  0.6595744680851063
40      50      Classification accuracy:  0.6595744680851063
40      60      Classification accuracy:  0.6595744680851063
40      70      Classification accuracy:  0.6595744680851063
40      80      Classification accuracy:  0.6595744680851063
40      90      Classification accuracy:  0.6595744680851063
40      100     Classification accuracy:  0.6595744680851063
50      1       Classification accuracy:  0.6595744680851063
50      10      Classification accuracy:  0.6595744680851063
50      20      Classification accuracy:  0.6595744680851063
50      30      Classification accuracy:  0.6595744680851063
50      40      Classification accuracy:  0.6595744680851063
50      50      Classification accuracy:  0.6595744680851063
50      60      Classification accuracy:  0.6595744680851063
50      70      Classification accuracy:  0.6595744680851063
50      80      Classification accuracy:  0.6595744680851063
50      90      Classification accuracy:  0.6595744680851063
50      100     Classification accuracy:  0.6595744680851063
60      1       Classification accuracy:  0.6595744680851063
60      10      Classification accuracy:  0.6595744680851063
60      20      Classification accuracy:  0.6595744680851063
60      30      Classification accuracy:  0.6595744680851063
60      40      Classification accuracy:  0.6595744680851063
60      50      Classification accuracy:  0.6595744680851063
60      60      Classification accuracy:  0.6595744680851063
60      70      Classification accuracy:  0.6595744680851063
60      80      Classification accuracy:  0.6595744680851063
```

```
60      90      Classification accuracy:  0.6595744680851063
60      100     Classification accuracy:  0.6595744680851063
70      1       Classification accuracy:  0.6595744680851063
70      10      Classification accuracy:  0.6595744680851063
70      20      Classification accuracy:  0.6595744680851063
70      30      Classification accuracy:  0.6595744680851063
70      40      Classification accuracy:  0.6595744680851063
70      50      Classification accuracy:  0.6595744680851063
70      60      Classification accuracy:  0.6595744680851063
70      70      Classification accuracy:  0.6595744680851063
70      80      Classification accuracy:  0.6595744680851063
70      90      Classification accuracy:  0.6595744680851063
70      100     Classification accuracy:  0.6595744680851063
80      1       Classification accuracy:  0.6702127659574468
80      10      Classification accuracy:  0.6595744680851063
80      20      Classification accuracy:  0.6595744680851063
80      30      Classification accuracy:  0.6595744680851063
80      40      Classification accuracy:  0.6595744680851063
80      50      Classification accuracy:  0.6595744680851063
80      60      Classification accuracy:  0.6595744680851063
80      70      Classification accuracy:  0.6595744680851063
80      80      Classification accuracy:  0.6595744680851063
80      90      Classification accuracy:  0.6595744680851063
80      100     Classification accuracy:  0.6595744680851063
90      1       Classification accuracy:  0.6595744680851063
90      10      Classification accuracy:  0.6595744680851063
90      20      Classification accuracy:  0.6595744680851063
90      30      Classification accuracy:  0.6595744680851063
```

```
90      40      Classification accuracy:  0.6595744680851063
90      50      Classification accuracy:  0.6595744680851063
90      60      Classification accuracy:  0.6595744680851063
90      70      Classification accuracy:  0.6595744680851063
90      80      Classification accuracy:  0.6595744680851063
90      90      Classification accuracy:  0.6595744680851063
90      100     Classification accuracy:  0.6595744680851063
100     1       Classification accuracy:  0.6595744680851063
100     10      Classification accuracy:  0.6595744680851063
100     20      Classification accuracy:  0.6595744680851063
100     30      Classification accuracy:  0.6595744680851063
100     40      Classification accuracy:  0.6595744680851063
100     50      Classification accuracy:  0.6595744680851063
100     60      Classification accuracy:  0.6595744680851063
100     70      Classification accuracy:  0.6595744680851063
100     80      Classification accuracy:  0.6595744680851063
100     90      Classification accuracy:  0.6595744680851063
100     100     Classification accuracy:  0.6595744680851063
```

Приложение 2.

```
Classification accuracy: 0.8554216867469879
Classification accuracy: 0.8554216867469879
Classification accuracy: 0.8674698795180723
Classification accuracy: 0.8554216867469879
Classification accuracy: 0.8554216867469879
Classification accuracy: 0.8554216867469879
Classification accuracy: 0.8554216867469879
Classification accuracy: 0.8554216867469879
```

```
Classification accuracy: 0.8554216867469879
Classification accuracy: 0.8674698795180723
Classification accuracy: 0.8554216867469879
Classification accuracy: 0.8554216867469879
Classification accuracy: 0.8554216867469879
Classification accuracy: 0.8554216867469879
Classification accuracy: 0.8674698795180723
Classification accuracy: 0.8554216867469879
Classification accuracy: 0.8554216867469879
Classification accuracy: 0.8554216867469879
Classification accuracy: 0.8433734939759037
Classification accuracy: 0.8674698795180723
```

Приложение 3.

```
Classification accuracy: 0.7297297297297297
y_test    predictions     real val.
-1          1.0          [0.06, 0.06]
Equal.           [0.51, 0.51]
Equal.           [10.2, 10.2]
-1          1.0          [0.06, 0.06]
Equal.           [0.08, 0.08]
Equal.           [0.04, 0.04]
Equal.           [2.87, 2.87]
Equal.           [0.25, 0.25]
Equal.           [3.23, 3.23]
Equal.           [3.88, 3.88]
```

```
1          -1.0          [0.03, 0.03]
Equal.           [0.08, 0.08]
-1          1.0          [1.44, 1.44]
Equal.           [0.14, 0.14]
Equal.           [0.52, 0.52]
-1          1.0          [0.24, 0.24]
Equal.           [0.11, 0.11]
Equal.           [0.12, 0.12]
-1          1.0          [0.16, 0.16]
Equal.           [0.57, 0.57]
Equal.           [1.47, 1.47]
Equal.           [0.7, 0.7]
Equal.           [0.16, 0.16]
Equal.           [0.01, 0.01]
Equal.           [0.01, 0.01]
Equal.           [0.04, 0.04]
Equal.           [0.04, 0.04]
-1          1.0          [0.06, 0.06]
Equal.           [0.12, 0.12]
Equal.           [0.08, 0.08]
-1          1.0          [0.06, 0.06]
Equal.           [0.21, 0.21]
Equal.           [0.35, 0.35]
Equal.           [68.1, 68.1]
1          -1.0          [3.2, 3.2]
Equal.           [0.92, 0.92]
Equal.           [5.89, 5.89]
Equal.           [0.58, 0.58]
```

```
Equal.           [0.008, 0.008]
Equal.           [0.03, 0.03]
Equal.           [0.007, 0.007]
Equal.           [0.03, 0.03]
Equal.           [0.9, 0.9]
Equal.           [0.4, 0.4]
1          -1.0          [0.8, 0.8]
Equal.           [0.1, 0.1]
1          -1.0          [0.2, 0.2]
-1          1.0          [0.15, 0.15]
-1          1.0          [0.13, 0.13]
1          -1.0          [0.04, 0.04]
Equal.           [0.32, 0.32]
Equal.           [0.33, 0.33]
Equal.           [0.66, 0.66]
Equal.           [0.03, 0.03]
Equal.           [0.11, 0.11]
Equal.           [0.17, 0.17]
Equal.           [0.09, 0.09]
1          -1.0          [1.04, 1.04]
Equal.           [0.17, 0.17]
-1          1.0          [0.12, 0.12]
-1          1.0          [0.33, 0.33]
Equal.           [0.08, 0.08]
Equal.           [0.11, 0.11]
Equal.           [0.96, 0.96]
Equal.           [0.17, 0.17]
Equal.           [0.03, 0.03]
```

```
Equal.            [0.02, 0.02]
Equal.            [0.03, 0.03]
Equal.            [0.01, 0.01]
1          -1.0          [0.003, 0.003]
-1          1.0          [0.02, 0.02]
Equal.            [0.03, 0.03]
1          -1.0          [0.04, 0.04]
Equal.            [0.03, 0.03]
```