Содержание:

[Исследование предметной области 2](#_Toc500859165)

1. [1.1 Теоретические основы разработки программы калькулятора с рукописным вводом 2](#_Toc500859166)

[1.1.1 Способы обучения искусственных систем 2](#_Toc500859167)

[1.1.2 Классификация задач индуктивного обучения 2](#_Toc500859168)

[1.1.3 Задача классификации для рукописного калькулятора 3](#_Toc500859169)

1. [1.2 Понятия рукописного калькулятора, его характеристика и классификации, обзор существующих решений и алгоритмов 3](#_Toc500859170)

[1.2.1 Понятие рукописного калькулятора и его характеристики 3](#_Toc500859171)

[1.2.2 Классификация рукописных калькуляторов 4](#_Toc500859172)

[1.2.3 Обзор существующих решений 4](#_Toc500859173)

[1.2.4 Алгоритмы для реализации распознавания рукописного текста: виды и особенности 5](#_Toc500859174)

[1.2.4.1 Мультиномиальная логистическая регрессия 6](#_Toc500859175)

[1.2.4.2 Наивный Байесовский классификатор 8](#_Toc500859176)

[1.2.4.3 Метод k ближайших соседей 11](#_Toc500859177)

[1.2.4.4 Метод Парзеновского окна 13](#_Toc500859178)

[1.2.4.5 Метод потенциальных функций 15](#_Toc500859179)

[1.2.4.6 Метод опорных векторов 18](#_Toc500859180)

1. [1.3 Вывод о выборе алгоритма классификации 24](#_Toc500859181)

[Литература: 25](#_Toc500859182)

Разработка программы калькулятора с рукописным вводом

# Исследование предметной области

## 1.1 Теоретические основы разработки программы калькулятора с рукописным вводом

Создание рукописного калькулятора невозможно без понятия “машинное обучение”. Машинное обучение - процесс, в результате которого машина (компьютер) способна показывать поведение, которое в нее не было явно заложено (запрограммировано)[1, C. 4].

### 1.1.1 Способы обучения искусственных систем

Способы обучения:

* Дедуктивное, аналитическое, обучение (экспертные системы)

Имеются формализованные и сформулированные экспертом знания.

* Индуктивное обучение (статическое обучение)

Имеются эмперические данные, на основе которых строится общее правило. Кроме того, эмпирические данные могут быть получены из предыдущих сеансов работы с программой.

* Комбинированное обучение

Задача реализации механизма распознавания цифр и математических знаков в рукописном калькуляторе принадлежит к классу задач индуктивного обучения.

### 1.1.2 Классификация задач индуктивного обучения

Задачи индуктивного обучения можно классифицировать следующим образом:

* Обучение с учителем
  + Классификация
  + восстановление регрессии
  + другие задачи
* Обучение без учителя
  + Кластеризация
  + понижение размерности
  + другие задачи
* Обучение с подкреплением
* Активное обучение
* другие виды обучения

### 1.1.3 Задача классификации для рукописного калькулятора

Для распознавания рукописного текста используется обучающая выборка имеющихся рукописных цифр, и на основе этой выборки производится обучение программы калькулятора. Опираясь на обучающую выборку, из входного рукописного символа на выходе должен получиться символ в машинном коде . Таким образом, задача создания механизма распознавания рукописных цифр сводится к задаче классификации обучения с учителем.

Рассмотрим подробнее обучение с учителем:

Множество X - исходная выборка примеров, множество Y - ответы на исходную выборку.

Имеется некоторая зависимость (детерминированная или вероятностная), позволяющая по xX предугадать(или оценить вероятность) yY. Если зависимость детерминированная, то существует )

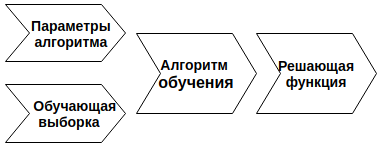
Зависимость известна только на объектах из обучающей выборки:

…,

Пара - прецедент.

Задача обучения с учителем: восстановить зависимость, т. е. найти функцию , предсказывающую по ответы . Т.е. для детерминированной зависимости найти

Наглядно схема обучения с учителе работает следующим образом:



[1, C.18-26]

## 1.2 Понятия рукописного калькулятора, его характеристика и классификации, обзор существующих решений и алгоритмов

### 1.2.1 Понятие рукописного калькулятора и его характеристики

Рукописный калькулятор - удобный инструмент для работы с вычислениями математических выражений при помощи рукописного ввода.

Основные умения данного приложения включают в себя: распознавание введённого рукописного текста и вычисление его значения.

### 1.2.2 Классификация рукописных калькуляторов

На сегодняшний день существует множество гаджетов, посредством которых можно использовать рукописный калькулятор.

Таким образом, можно выделить следующие реализации рукописного калькулятора:

* мобильное приложение для:
  + iOS (MyScript Calculator)
  + Android (MyScript Calculator)
* для настольных устройств под:
  + Windows
  + Linux
  + macOS (Touch-Calculator)
* web-версия (Web-версия Mathematical Expression Recognition)

### 1.2.3 Обзор существующих решений

* MyScriplculator

На сегодняшний день этот калькулятор является одним из лучших: он распознает математические выражения целиком, знает множество математических функций, общепринятых обозначений и даже умеет искать недостающие части уравнений.

Существует версия для Android и iOS. Однако те, кому было бы удобно работать на своем ПК с рукописным калькулятором, с легкостью могут решить проблему отсутствия реализации настольной версии при помощи виртуальной машины. Еще одним приятным бонусом является то, что приложение бесплатно.

* Touch-Calculator

Реализация калькулятора с рукописным вводом под систему MacOS. Калькулятор имеет как интерфейс с “кнопочными” цифрами и математическими знаками, так и поле для рукописного ввода. Распознавание проходит достаточно точно и быстро. Её единственный недостаток - распознавание происходит только по одному символу.

* Web-версия Mathematical Expression Recognition

Работает не точно и неудобна в использовании, а также не производит вычислений

Таким образом, существуют удобные и качественные реализации для мобильных устройств, которые можно использовать как настольные версии при помощи виртуальной машины. У web-версии есть реализация, однако она обладает следующими негативными свойствами: интерфейс не стилизован, низкое качество классификации, не производит вычисление введенного выражения . Из обзора существующих решений задачи можно сделать вывод о том, что самым актуальным решением будет создание web-версии калькулятора, который распознает математическое выражение и вычисляет его.

### 1.2.4 Алгоритмы для реализации распознавания рукописного текста: виды и особенности

Для реализации распознавания символов необходимо использовать алгоритмы решения задач многоклассовой классификации.

Введем общие обозначения:

Множество X — объекты, примеры, входы.

Множество Y — ответы, отклики, «метки», выходы.

N - длина тренировочной выборки.

K - количество классов.

Пара (xi, yi) ∈ X × Y — прецедент.

xi - признаковое описание объекта x, то есть представляет собой вектор размерности k, который содержит вероятности, с которыми данный объект принадлежит каждому из классов.

yi - вектор размерности k, в котором элемент, являющийся правильным ответом содержит 1, а все остальные элементы заполнены 0.

{(xi, yi), …, (xN, yN) - тренировочная выборка, где .

**Оценки точности:**

Для оценки точности алгоритмов классификации исходная выборка разбивается на тренировочную выборку и тестовую. На тренировочной выборке происходит обучение модели, а на тестовой происходит проверка точности. Точностью данного алгоритма будет являться процент правильно классифицированных объектов из тестовой выборки, полагаясь на параметры, полученные при помощи обучения на тренировочной выборке.

#### 1.2.4.1 Мультиномиальная логистическая регрессия

**Описание**:

Мультиномиальная логистическая регрессия позволяет классифицировать объекты по нескольким классам (в том числе более, чем двум), используя за основу логистическую регрессию. Таким образом, данный алгоритм позволяет решить проблему многоклассовой классификации. Данная модель прогнозирует вероятность возможных исходов, опираясь на исходный набор независимых переменных. Алгоритм используется в тех случаях, когда переменная является категориальной, что означает, что она может принадлежать лишь одной из категорий, которые не связаны между собой. [2]

**Математическая модель:**

Данная модель предполагает, что на входе имеется множество объектов X, которому соответствует множество ответов Y. Имея на входе эти данные предполагается некоторая гипотеза, которая оценивает вероятность P = (y = k|x), где k = 1, …, K, то есть вероятность, с которой каждый объект исходного множества принадлежит каждому из K классов.

Тогда гипотеза принимает вид вектора размерностью K, который будет содержать в себе вероятности попадания объекта в K различных классов. Сумма элементов данного вектора должна давать 1. Таким образом, в математическом виде гипотеза имеет следующий вид:

=

Здесь w1, w2, …, wK - параметры модели. Стоит отметить, что в основе данной гипотезы лежит нормальное распределение.

Для того, чтобы обозначить все параметры модели, определяется матрица W размерности N x K, то есть содержит параметры модели для каждого элемента исходной выборки.

Далее необходимо определить параметры модели. Подбор параметром в данной модели осуществляется при помощи минимизации некоторой функции стоимости, которая описывается следующим образом:

При использовании мультиномиальной регрессии вероятность попадания объекта в класс определяется следующим образом:

Для минимизации функции необходимо рассчитать градиент и при его помощи минимизировать функцию при помощи какого-либо метода минимизации.

После того, как функция минимизирована, получаем параметры математической модели, которые будут использованы для определения гипотезы. Используя эту гипотезу и параметры входного тестового объекта определяется его класс. [2]

**Псевдокод:**

**Вход:**

K - количество классов

N - длина тренировочной выборки

M - количество признаков у объектов

x - тренировочная выборка объектов размером NxM

y - ответы к тренировочной выборке размером N

**Выход:**

class - класс объекта

**Псевдокод:**

theta = rand[K][M] - параметр для классификации

cost := 0

ДЛЯ КАЖДОГО n из N

sum := 0

ДЛЯ КАЖДОГО k∊K

sum := sum + exp(theta[k]T \* x[n])

ДЛЯ КАЖДОГО k∊K

ЕСЛИ (y[n] = k)

then cost := cost - log((exp(theta[k]T \* x[n]))/(sum))

cost := cost / N

w = lambd \* ||theta||2

cost := cost + w

ПОКА (cost - не минимизирована)

gradient = lambda \* theta[k]

ДЛЯ КАЖДОГО n∊N

ЕСЛИ (y[n] = k)

then gradient = gradient - x[n] \* (1 - (exp(theta[k]T \* x[n]))/sum

gradient := gradient / N

theta[k] := результат следующего шага минимизации функции cost

obj = arr[m]

class = k[0]

ДЛЯ КАЖДОГО k∊K

ЕСЛИ ((exp(theta[k]T \* x[n]))/(sum) > (exp(theta[class]T \* x[n]))/(sum))

ТОГДА class := k

**Преимущества и недостатки:**

*Параметры*: w

*Скорость обучения*: 110 секунд

*Необходимость переобучения*: обучение требуется тодин раз, затем используется полученный в процессе обучения параметр w.

*Скорость классификации 10000 объектов:* ≈ 0.04 секунды

*Качество классификации*: ≈ 92.54%

*Объём занимаемой памяти*: Нет необходимости хранить выборку, так как для классификации данным методом необхадомо хранить единтсвенный параметр w.

*Выбор параметров*: данный алгоритм не содержит параметров, которые вычисляются или задаются каким-либо образом для улучшения качества классфикации, однако алгоритм позволяет использовать различные методы минимизации функции для подбора наилучшего параметра w (например, метод градиентного спуска, метод Ньютона, Квазиньютоновские методы и т.д.)

#### 1.2.4.2 Наивный Байесовский классификатор

**Описание:**

Наивный Байесовский классификатор позволяет осуществить многоклассовую классификацию объектов, опираясь главным образом на теорему Байеса. Данная теорема предполагает расчет вероятности того, что данная причина привела наблюдаемому событию, на основе знаний о том, с какой вероятностью причина приводит к некоторому событию.

Так как при классификации объектов ответом существует только один правильный ответ, поэтому используется не вероятность, с которой объект принадлежит каждому классу, а наиболее вероятный из классов.

Наивность классификатора заключается в том, что “наивно” предполагается вероятность принадлежности объекта к некоторому классу, представляя объект, как набор признаков, вероятности которых условно никак не зависят друг от друга. [3, 4, 5]

**Математическая модель:**

В основе данной модели лежит теорема Байеса:

,

где

P(k|x) - вероятность того, что объект х принадлежит классу k,

P(x|k) - вероятность того, что объект х встречается среди объектов класса k,

P(k) - безусловная вероятность встретить объект класса k,

P(x) - безусловная вероятность объекта x.

Далее необходимо определить, к какому классу принадлежит объект. Определение объекта к конкретному классу Байессовский классификатор осуществляет посредством оценки априорного максимума.

Так как вероятность P(x) для всех объектов одинакова, то формулу можно записать в следующем виде:

Для вычисления максимально-вероятностного класса необходимо рассчитать вероятность того, что объект x принадлежит классу k.

Объект представляется в виде набора признаков, вероятности которых условно не зависят друг от друга:

Теперь Наивный Байессовский классификатор приобретает вид:

При большом количестве признаков происходит многократное перемножение чисел меньше единицы, что может выйти за границы арифметического переполнения, производить расчеты наиболее безопасно в логарифмическом пространстве:

Безусловная вероятность встретить объект класса k оценивается по тренировочной выборке:

,

Nk - количество объектов в тренировочной выборке, принадлежащих классу k

N - количество всех объектов в тренировочной выборке

Оценка параметров Байесовской модели

,

где

wik - общее количество элементов с заданным значением признака i в классе k

a - параметр, значение которого всегда больше 0, вводится для того, чтобы значение вероятности не принимало нулевое значение.

Таким образом, конечный вид наивного Байесовского классификатора имеет следующий вид:

**Псевдокод:**

**Вход:**

K - количество классов

N - длина тренировочной выборки

M - количество признаков у объектов выборки

x - тренировочная выборка объектов размером NxM

y - ответы к тренировочной выборке размером N

x’- классифицируемый объект размером M

**Выход:**

class - класс объекта

**Псевдокод:**

N\_k = arr[K]

W\_m = arr[M] - набор признаков

ДЛЯ КАЖДОГО k∊K

N\_k[k] = 0

ДЛЯ КАЖДОГО m∊M

W\_m[m] = 0

ДЛЯ КАЖДОГО n∊N

ДЛЯ КАЖДОГО k∊K

if (y[n] = k)

then N\_k[k] := N\_k[k] + 1

ДЛЯ КАЖДОГО m∊M

if (y[n] = k)

then N\_k[k] := N\_k[k] + 1

class = k[0]

ДЛЯ КАЖДОГО k∊K

P\_k := N\_k[k] / N

sum := 0

ДЛЯ КАЖДОГО m∊M

sum := sum + x[m]

**Преимущества и недостатки:**

*Скорость обучения*:

*Качество классификации*:

*Объём занимаемой памяти*: умеренные, нет необходимости хранить всю выборку

*Параметры*: а

*Выбор параметров*: выбор параметра а практически не влияет на качество классификации, он задаётся с целью избежать нулевых значений

#### 1.2.4.3 Метод k ближайших соседей

**Описание:**

Данный метод относит классифицируемый объект к классу, которому принадлежит большинство из его ближайших соседей. Ближайшие соседи определяются по имеющимся у объектов признакам. При обучении алгоритм запоминает все признаки и то, каким классам они соответствуют. При появлении объекта, который необходимо классифицировать, вычисляется расстояние между значениями каждого из признаков. После того, как эти значения вычислены, по каждому признаку выбирается ближайший сосед. [6, 7]

**Математическая модель:**

Для начала необходимо выбрать количество ближайших соседей k, по которым будет происходить оценка объекта, класс которого необходимо определить.

Если взять малое значение k, то появится большое количество выбросов, что снизит качество работы алгоритма, а большое значение может значительно увеличить сложность вычислений и исказить логику ближайших соседей, так как есть вероятность того, что ближайшие точки принадлежат одному классу, в соответствии с гипотезой компактности.Чаще всего значение k определяют по критерию скользящего контроля, а именно методом исключения по одному.

Все объекты тренировочной выборки располагаются в следующей последовательности:

,

где - функция расстояния.

Функция расстояния может быть задана множеством способов, используем, например, Евклидово расстояние:

Из полученной последовательности необходимо выбрать k’ первых элементов, по которым будет определяться принадлежность классифицируемого объекта к какому-либо классу.

Для каждого выбранного объекта из тренировочной выборки определяется класс, к которому он принадлежит

Пусть нужно определить класс для некоторого объекта z. Тогда функция для классификации объекта z будет выглядеть следующим образом:

где wiz - вес i-го объекта из упорядоченной по расстоянию тренировочной выборки для объекта z. [6, 7]

**Псевдокод:**

**Вход:**

K - количество классов

N - длина тренировочной выборки

M - количество признаков у объектов

C - количество ближайших соседей

x - тренировочная выборка объектов размером NxM

y - ответы к тренировочной выборке размером N

x’- классифицируемый объект размером M

**Выход:**

class - класс объекта

**Псевдокод:**

neighbours := sort(x, x’) - сортировка объектов тренировочной выборки по расстоянию до классифицируемого объекта

weight := arr[C]

i := C

ДЛЯ КАЖДОГО с∊C

weight[c] := i

i := i - 1

class := K[0]

w\_class := 0

ДЛЯ КАЖДОГО k∊K

w := 0

ДЛЯ КАЖДОГО c∊C

if (neighbours[c] = k)

then w := w + weight[c]

ЕСЛИ (w > w\_class)

ТОГДА

w\_class = w

class = k

**Преимущества и недостатки:**

*Параметры*: k,p

*Скорость обучения*: 110 секунд

*Необходимость переобучения*: процесс обучения необходим для классификации каждого нового объекта

*Скорость классификации 10000 объектов:* ≈ 0.04 секунды

*Качество классификации*: ≈ 92.54%

*Объём занимаемой памяти*: необходимо хранить всю выборку. База данных MNIST занимает приблизительно 12 Мб. К этому объему памяти также добавится еще объём базы данных для математических символов и память, необходимая для работы программы

*Выбор параметров*:

* k – количество ближайших соседей, по которым происходит оценка

Выбор наиболее оптимального параметра k может быть выбран различными способами. Наиболее оптимальный вариант можно подобрать, воспользовавшись, например, методом скользящего контроля.

* р – функция расстояния

Можно использовать различные функции для вычисления расстояний в многомерном пространстве

#### 1.2.4.4 Метод Парзеновского окна

**Описание:**

Метод k ближайших соседей лежит в основе метода Парзеновского окна. Для каждого из объектов тренировочной выборки определяется расстояние до классифицируемого объекта по имеющимся признакам. В методе k ближайших соседей по некоторым критериям выбиралась фиксированная величина количества соседей, которые играют роль в определении класса некоторого объекта, и их веса, вычисляемые по определённой функции, зависящей от ранга соседа. В методе Парзеновского окна соседи, влияющие на классификацию, определяются теми объектами тренировочной выборки, которые попали в окрестность классифицируемого объекта. Ширина окрестности задаётся некоторым параметром h (в методе ближайших соседей подобным параметром был k). Вместо весов объектов для данного алгоритма задается некоторая функция, которая зависит не от ранга соседа, а от расстояния от классифицируемого объекта до выбранного соседа из тренировочной выборки. [7]

**Математическая модель:**

Используя алгоритм ближайших соседей, первоначально необходимо выбрать количество ближайших соседей k, по которым будет происходить оценка объекта, класс которого необходимо определить. В данном алгоритме по подобной логике выбирается ширина Парзеновского расстояния.

Если взять малое значение h, то очень маленькое окно может привести к неустойчивой классификации, а большое окно приводит к вырождению алгоритма в константу. Как и для параметра k из метода ближайших соседей, чаще всего значение h определяют по критерию скользящего контроля, а именно методом исключения по одному.

Все объекты тренировочной выборки располагаются в последовательности, располагающейся по возрастанию расстояний до объектов:

,

где - функция расстояния.

Функцию расстояния можно выбрать любой.

Пусть нужно определить класс для некоторого объекта z. Тогда функция для классификации объекта z методом окна Парзена будет выглядеть следующим образом:

где K - ядро.

Ядро может быть выбрано из следующего набора ядер:

* Епанечникова
* Квартическое
* Треугольное
* Гауссовское
* Прямоугольное

Выбор наилучшей функции для вычисления ядра невозможно предсказать. Наиболее эффективную функцию можно выбрать лишь воспользовавшись экспериментальным путем.

К не сильно влияет на качество классификации, однако она определяет, насколько гладкой будет функция распределения плотностей. Также вид выбранного ядра может повлиять на эффективности вычислений. [7]

**Псевдокод:**

**Вход:**

K - количество классов

N - длина тренировочной выборки

M - количество признаков у объектов

h - ширина окна

x - тренировочная выборка объектов размером NxM

y - ответы к тренировочной выборке размером N

x’- классифицируемый объект размером M

**Выход:**

class - класс объекта

**Псевдокод:**

neighbours := sort(x, x’)

I := 0

ПОКА (distanse (neighbours[I], x’) < h)

I := I + 1

kernel := arr[I]

ДЛЯ КАЖДОГО i∊I

kernel[i] := K(distance(neighbours[i],x’)/h) - функция ядра

class := K[0]

w\_class := 0

ДЛЯ КАЖДОГО k∊K

w := 0

for i from I

ЕСЛИ (neighbours[i] = k)

ТОГДА w := w + kernel[i]

ЕСЛИ (w > w\_class)

ТОГДА

w\_class := w

class := k

**Преимущества и недостатки:**

*Параметры*: h, p, K

*Скорость обучения*: 110 секунд

*Необходимость переобучения*: процесс обучения необходим для классификации каждого нового объекта

*Скорость классификации 10000 объектов:* ≈ 0.04 секунды

*Качество классификации*: ≈ 92.54%

*Объём занимаемой памяти*: необходимо хранить всю выборку. База данных MNIST занимает приблизительно 12 Мб. К этому объему памяти также добавится еще объём базы данных для математических символов и память, необходимая для работы программы

*Выбор параметров*:

* h – ширина окна, в пределах которого выбираются соседи, по которым происходит оценка

Выбор наиболее оптимального параметра h может быть выбран различными способами. Наиболее оптимальный вариант можно подобрать, воспользовавшись, например, методом скользящего контроля.

* р – функция расстояния

Можно использовать различные функции для вычисления расстояний в многомерном пространстве

* K – функция ядра

Можно использовать различные функции для определения ядра

#### 1.2.4.5 Метод потенциальных функций

**Описание:**

В основе этого алгоритма лежит метод ближайших соседей. Метод потенциальных функций оценивает важность объектов обучающей выборки при задаче классификации нового объекта.

Основная идея этого метода пришла из такого определения, как потенциал электростатического поля элементарной заряженной частицы. Проводится аналогия с потенциалом электростатического поля, который прямо пропорционален заряду частицы, порождающей это поле и обратно пропорционален расстоянию от точки в пространстве, в которой мы пытаемся вычислить потенциал, до заряженной частицы.

Таким образом для реализации алгоритма потенциальных функций выглядит следующим образом: класс(потенциал) некоторого объекта (точки в пространстве) прямо пропорционально зависит от степени важности (величины заряда) объектов обучающей выборки (всех заряженных частиц, влияющих на потенциал) и обратно пропорционально расстоянию до них. [7,8]

**Математическая модель:**

Итак, как и в методе ближайших соседей, все объекты обучающей выборки располагаются в порядке возрастания расстояния до классифицируемого объекта:

,

где - функция расстояния.

Алгоритм классификации методом потенциальных функций выглядит следующим образом:

где wiz - степень важности (то есть вес) i-го объекта обучающей выборки, расположенной в порядке возрастания расстояния до объекта.

Особенностью метода потенциальных функций является способ выбора функции определения веса объекта обучающей выборки. Вес находится следующим образом:

,

где

, где а - константа, введенная для того, чтобы избежать нулевого значения в знаменателе функции К,

- параметр, который задаёт вес объекта,

- значение расстояния от z до i-го ближайшего к z объекта ,

- радиус действия объекта, который вводится также, как ширина окна Парзена. [7,8]

**Псевдокод:**

**Вход:**

K - количество классов

N - длина тренировочной выборки

M - количество признаков у объектов

h - ширина окна

x - тренировочная выборка объектов размером NxM

y - ответы к тренировочной выборке размером N

x’- классифицируемый объект размером M

**Выход:**

class - класс объекта

**Псевдокод:**

a := 1

neighbours := sort(x, x’)

I := 0

ПОКА (distance (neighbours[I], x’) < h)

I := I + 1

weight := arr[I]

param := arr[N]

ДЛЯ КАЖДОГО n∊N

param[n] := 1

ДЛЯ КАЖДОГО i∊I

ПОКА (Q(classifier,X) > epsilon) - общий заряд всей выборки должен быть стабилизирован

ЕСЛИ (classifier(x[i], param[i]) != y[i])

ТОГДА param[i] := param[i] + 1

function classifier(x’, param)

ДЛЯ КАЖДОГО i∊I

weight[i] := K(distance(neighbours[i],x’)/h) - функция ядра

weight[i] := 1/(weight[i] + a)

weight[i] := param[i] \* weight[i]

class := K[0]

w\_class := 0

ДЛЯ КАЖДОГО k∊K

w := 0

ДЛЯ КАЖДОГО i∊I

ЕСЛИ (neighbours[i] = k)

ТОГДА w := w + weight[i]

ЕСЛИ (w > w\_class)

ТОГДА

w\_class := w

class := k

ВЕРНУТЬ class

**Преимущества и недостатки:**

*Скорость обучения*:

*Качество классификации*:

*Объём занимаемой памяти*:

*Параметры*: h, p,

*Выбор параметров*:

* h – ширина окна, в пределах которого выбираются соседи, по которым происходит оценка

Выбор наиболее оптимального параметра h может быть выбран различными способами. Наиболее оптимальный вариант можно подобрать, воспользовавшись, например, методом скользящего контроля.

* р – функция расстояния

Можно использовать различные функции для вычисления расстояний в многомерном пространстве

* – параметр, который задаёт вес объекта

Наиболее качественный параметр можно определить практически для конкретного примера

#### 1.2.4.6 Метод опорных векторов

**Описание:**

Данный метод первоначально предназначен для бинарной классификации, однако существует и модификация для многоклассовой. Суть метода заключается в том, что все объекты тренировочной выборки представлены в некотором k-мерном пространстве в виде вектора размерности k. Для разделения имеющихся объектов в пространстве используется так называемая плоскость классификатора, которая представляет собой гиперплоскость размерностью k-1. Логично, что таких плоскостей можно провести бесконечно большое количество. В алгоритме опорных векторов лучшей разделяющей плоскостью считается плоскость, расстояние от которой до каждого из классов максимально. В итоге пространство оказывается разделено на участки, каждый из которых соответствует какому-либо классу.

После того как плоскость проведена определяется положение каждого нового объекта ( то есть объекта, для которого необходимо решить задачу классификации) и ему присваивается тот класс, в участок которого попал этот объект. [9, 10, 11, 12]

**Математическая модель:**

Для начала опишем математическую модель работы алгоритма для двух классов. То есть ответы для выборки значений Х принимают значения Y={1, -1}.

Будем считать, что все ответы множества ответов Y могут принимать только одно из двух значений: k1 или k2.

Данный алгоритм подразумевает создание некоторой линейной функции , удовлетворяющей следующим условиям:

,

где w - нормальный вектор к разделяющей гиперплоскости, <w,x> - скалярное произведение, b - некоторый параметр, называемый скалярным порогом. Такая функция позволяет задавать любую гиперплоскость в виде для некоторых w и b, которая будет реализовывать разбиение выборки по классам.

Однако нетрудно заметить, что можно построить бесконечно большое количество таких разделяющих плоскостей. Суть данного метода заключается в том, чтобы построить наиболее оптимальную разделяющую плоскость. Данный алгоритм предполагает, что наиболее оптимальной будет являться та разделяющая гиперплоскость , которая стоит максимально далеко от тех точек обоих классов, которые лежат ближе всего к этой плоскости.

Тогда задача сводится к тому, чтобы подобрать такие параметры w и b, которые будут максимизировать расстояние от ближайших точек каждого класса до разделяющей гиперплоскости.

Будем считать, функция f(x) принимает значения от -1 до 1, т.е.:

. Это условие задаёт полосу, разделяющую два класса. В пределах данной полосы не может находиться ни один объект из выборки. Две параллельные плоскости с направляющим вектором w являются границами полосы, а оптимальная разделяющая гиперплоскость лежит ровно посередине двух граничных плоскостей.

Значит, для того, чтобы найти максимально отдалённую от точек каждого класса плоскость, необходимо определить максимальную ширину полосы. Ширина этой полосы примет значение. Тогда расстояние от ближайших точек двух классов до оптимальной разделяющей гиперплоскости примет значение половины ширины полосы, то есть . Нахождение максимума расстояния от ближайших точек до плоскости эквивалентно нахождению минимума

С учётом того, что если ответ принимает значение -1 или 1, то функция принимает значения -1 и 1 соответственно, задача сводится к решению следующей системы уравнений:

Используем теорему Куна-Таккера. Тогда систему уравнений можно свести к двойственной задаче поиска седловой точки функции Лагранжа:

Здесь - вектор двойственных переменных.

Далее полученную систему уравнений можно свести к эквивалентной задаче квадратичного программирования, содержащую только двойственные переменные:

После решения данной системы можно найти w и b:

Тогда функция классификации будет выглядеть следующим образом:

На практике не все выборки можно разделить линейной гиперплоскостью. В случае, когда необходимо разделить линейно неразделимую выборку, все элементы этой выборки вкладываются в некоторое пространство, размерность которого выше, чем размерность заданной выборки, при помощи некоторого отображения Это отображение выбирается таким образом, чтобы в новом пространстве выборка была линейно-разделимой.

Для того, чтобы поставить задачу в данном случае по аналогии с задачей для линейно-разделимой выбоки, вводится набор дополнительных переменных , которые являются величиной ошибки на объектах тренировочной выборки X. Необходимо ввести суммарную ошибку классификации, которую необходимо минимизировать:

Здесь C является параметром настройки метода. Посредством использования этого параметра можно регулировать отношение между максимизацией ширины разделяющей полосы и минимизацией суммарной ошибки.

Также, как и для линейно-разделимой выборки, воспользовавшись теоремой Куна-Таккера, сведем поставленную задачу к поиску седловой точки функции Лагранжа:

Эта задача также, как и в предыдущем случае, сводится к эквивалентной задаче двойственного программирования:

Далее по формулам линейно-разделимой выборки находятся w и b и определяется класс объекта.

Ядро классификатора имеет следующий вид:

Любая положительно определенная симметричная функция двух переменных может быть ядром. Наиболее оптимальные виды ядра, которые применяются чаще всего в задачах классификации:

* полиномиальное
* радиальная базисная функция
* Гауссова радиальная базисная функция
* сигмоид

Наиболее качественный вариант из предложенных или свой вариант ядра можно выявить экспериментальным путем для конкретной задачи.

С математической точки зрения ядром может служить любая положительно определенная симметричная функция двух переменных. Положительная определенность необходимо для того, чтобы соответствующая функция Лагранжа в задаче оптимизации была ограничена снизу, т.е. задача оптимизации была бы корректно определена

Необходимо преобразовать этот алгоритм бинарной классификации к алгоритму многоклассовой классификации.

Для этого используется правило, которое разбивает задачу на бинарные классы.В таком случае получаем, что если имеется k классов, то необходимо выделить k разбиений. Объект проверяется на принадлежность или непринадлежность лишь одному из классов. Если объект не является объектом этого класса, то используется разбиение, которое отделяет от остальных следующий класс. Такие действия происходят до тех пор, пока не найдётся тот класс, которому принадлежит объект. Функция классификации в это случае будет выглядеть следующим образом:

[9, 10, 11, 12]

**Псевдокод (для случая бинарного классификатора):**

**Вход:**

K - количество классов

N - длина тренировочной выборки

M - количество признаков у объектов

x - тренировочная выборка объектов размером NxM

y - ответы к тренировочной выборке размером N

x’- классифицируемый объект размером M

C - параметр двойственной задачи

lambd - вектор двойственных переменных размером N

**Выход:**

class - класс объекта

**Псевдокод:**

Q := arr[N][N]

ДЛЯ КАЖДОГО n∊N

ДЛЯ КАЖДОГО n’∊N

Q[n][n’] := y[n]y[n’]K(x[n][n’])

e := arr[N]

ДЛЯ КАЖДОГО n∊N

e[n] := 1

x\_o - множество периферийных объектов, lambd = 0

x\_s - множество опорных объектов 0 < lambd < C

x\_c - множество объектов-нарушителей, lambd = C

y\_o - ответы на объекты из множества x\_o

y\_s - ответы на объекты из множества x\_s

y\_c - ответы на объекты из множества x\_c

e\_o - объекты из е, которые соответствуют объектам из x\_o

e\_s - объекты из е, которые соответствуют объектам из x\_s

e\_c - объекты из е, которые соответствуют объектам из x\_c

lambd\_s - объекты из lambd, которые соответствуют объектам из x\_s

Q\_ss - множество объектов из Q, которые соответствуют парам объектов из x\_s

Q\_oo - множество объектов из Q, которые соответствуют парам объектов из x\_o

Q\_cc - множество объектов из Q, которые соответствуют парам объектов из x\_c

Q\_os и Q\_so - множество объектов из Q которые соответсвуют парам из x\_o, x\_s и x\_s, x\_o, соответственно

Q\_oc и Q\_co - множество объектов из Q которые соответсвуют парам из x\_o, x\_c и x\_c, x\_o, соответственно

Q\_cs и Q\_sc - множество объектов из Q которые соответсвуют парам из x\_c, x\_s и x\_s, x\_c, соответственно

ПОКА (∃i ∈ (x\_o U x\_c), которые необходимо перевести в x\_s)

ПОКА (∃i ∈ x\_s, которые необходимо перевести в x\_o или в x\_c)

решить оптимизационную задачу относительно lambd\_s:

ЕСЛИ (|x\_s|>2) И (∃i ∈ x\_s) И (lambd[i] <= 0)

ТОГДА перевести i в x\_o

ЕСЛИ (|x\_s|>2) И (∃i ∈ x\_s) И (lambd[i] >= C)

ТОГДА перевести i в x\_c

w := 0

ДЛЯ КАЖДОГО n∊N

w := w + lambd[i]\*y[i]\*x[i]

w\_0 := 0

w\_0\_count := 0

ДЛЯ КАЖДОГО n∊N

if (lambd[n] > 0)

w\_0 := w\_0 + (<w, x[i]> - y[i])

w\_0\_count := w\_0\_count + 1

w\_0 := w\_0 / w\_0\_count

m - множество отступов для объектов их x\_o и x\_c

ДЛЯ КАЖДОГО i∊(x\_o U x\_c)

m[i] := y[i](<w,x[i]> - w\_0)

ЕСЛИ (∃i ∈ x\_o) И (m[i] <= 1)

ТОГДА перевести i в x\_s

ЕСЛИ (∃i ∈ x\_c) И (m[i] >= 1)

ТОГДА перевести i в x\_s

sign := 0

ДЛЯ КАЖДОГО n∊N

sign := sign + (lambd[i]\*y[i]\*<x[i], x’> - w\_0)

ЕСЛИ (sign < 0)

ТОГДА class := class\_1

ИНАЧЕ class := class\_2

**Преимущества и недостатки:**

*Скорость обучения*:

*Качество классификации*:

*Объём занимаемой памяти*:

*Параметры*: h, p, K

*Выбор параметров*:

* h – ширина окна, в пределах которого выбираются соседи, по которым происходит оценка

Выбор наиболее оптимального параметра h может быть выбран различными способами. Наиболее оптимальный вариант можно подобрать, воспользовавшись, например, методом скользящего контроля.

* р – функция расстояния

Можно использовать различные функции для вычисления расстояний в многомерном пространстве

* K – функция ядра

Можно использовать различные функции для определения ядра

## 1.3 Вывод о выборе алгоритма классификации

Итак, в данной главе были выявлены основные преимущества и недостатки каждого из методов. Самым главным критерием для выбора алгоритма является качество классификации. Лучшими показателями обладает метод опорных векторов с использованием ядра. Кроме того, он обладает возможностями выбора различных параметров и функций ядра, также этот алгоритм предлагает способы для выбора самых оптимальных параметров. Недостатками метода являются низкая скорость обучения и объем занимаемой памяти, также недостатком является то, что необходимо провести обучение столько раз, сколько классов в выборке. Скорость обучения низкая, однако разделяющую гиперплоскость между каждым классом и остальными требуется “провести” только один раз. В алгоритме требуется хранить всю выборку, однако база MNIST занимает сравнительно небольшой объём памяти и не должна сильно сказаться на скорости работы классификатора. Решение проблемы скорости работы алгоритма при многоклассовой классификации, уже было предложено в работе Краммера и Зингера.

# Литература:

1. Н.Ю. Золотых. МАШИННОЕ ОБУЧЕНИЕ И АНАЛИЗ ДАННЫХ (Machine Learning and Data Mining)
2. <https://en.wikipedia.org/wiki/Multinomial_logistic_regression>
3. <http://bazhenov.me/blog/2012/06/11/naive-bayes.html>
4. <https://habrahabr.ru/post/120194/>
5. <http://www.machinelearning.ru/wiki/index.php?title=Наивный_байесовский_классификатор>
6. <http://www.machinelearning.ru/wiki/index.php?title=Метод_ближайшего_соседа>
7. Машинное обучение (Machine Learning) Метрические методы классификации и регрессии Уткин Л.В
8. http://www.machinelearning.ru/wiki/index.php?title=Метод\_потенциальных\_функций
9. <http://www.machinelearning.ru/wiki/index.php?title=SVM>
10. К.В. Воронцов “Лекции по методу опорных векторов”
11. Alexey Nefedov “Support Vector Machines: A Simple Tutorial”
12. Юрий Лифшиц “Алгоритмы для интернета. Метод опорных векторов”