УДК 7.90-2007

**ПОДХОДЫ К СОЗДАНИЮ КАЛЬКУЛЯТОРА С РУКОПИСНЫМ ВВОДОМ**

*С. А. Беляев, к.т.н., доцент, bserge@bk.ru*

*Д.А. Лапцевич, студентка кафедры МО ЭВМ,* *darya.laptsevich@gmail.com*

*Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина) (СПбГЭТУ «ЛЭТИ»), ул. Профессора Попова, 5, Санкт-Петербург, 197376*

В статье рассмотрены подходы к созданию калькулятора с рукописным вводом. Исследованы существующие решения, их достоинства и недостатки. На основе выявленных особенностей готовых решений предложено собственное решение, его математическая модель, архитектура и сценарии использования предлагаемого решения. Приведены результаты экспериментов для различных методов классификации рукописного текста, а также для параметров, улучшающих работу классификатора на базе данных образцов рукописного написания цифр MNIST. Приведены результаты экспериментов.

***Ключевые слова:*** *калькулятор с рукописным вводом, многоклассовая классификация, метод опорных векторов, MNIST*

***Введение***

На сегодняшний день техника окружает нас всюду, более того, её «интеллектуальные способности» прогрессируют с каждым годом. Главной задачей для разработчиков программного обеспечения является удобство и практичность в «общении» пользователя с техникой.

Одними из самых популярных средств коммуникации пользователя и компьютера являются голосовой ввод и ввод рукописного текста. Эти возможности стали нормой и функциональным инструментом для современного пользователя.

Практически каждый человек использует калькулятор. Любая сфера деятельности подразумевает некоторые расчеты: бухгалтер практически каждый день сталкивается с расчетами затрат фирмы, домохозяйка подсчитывает расходы на коммунальные услуги, студент вычисляет ответ для задачи в контрольной работе. В любом случае, каждая целевая аудитория сталкивается с вычислениями, которые довольно затруднительно произвести без помощи вычислительной машины.

Приложение калькулятора с рукописным или голосовым вводом было бы весьма востребовано среди пользователей, которые привыкли пользоваться данными инструментами. В данной статье рассмотрим вариант создания калькулятора с рукописным вводом.

1. ***Существующие решения***

В настоящее время существуют решения, предлагающие рукописный ааод формул, в некоторых случаях с последующим их вычислением:

* MyScript Calculator;
* Touch-Calculator;
* Mathematical Expression Recognition.

MyScript Calculator [1].

На сегодняшний день этот калькулятор является практически универсальным: он распознает математические выражения целиком, знает множество математических функций, общепринятых обозначений, распознает степени, индексы, дроби и даже умеет искать недостающие части уравнений.

Существуют версии для Android и iOS.

Touch-Calculator [2].

Реализация калькулятора с рукописным вводом для системы MacOS. Калькулятор имеет как интерфейс с “кнопочными” цифрами и математическими знаками, так и поле для рукописного ввода. Распознавание символа проходит за доли секунды, а ошибка классификации очень близка к нулю. Единственный недостаток данной реализации – это то, что распознавание происходит только по одному символу.

Mathematical Expression Recognition [3].

Классификация символов математического выражения происходит некачественно (из 100 введенных символов 21 символ был распознан некорректно, то есть ошибка классификации достигает 21%). Приложение осуществляет распознавание математического выражения без выведения результата вычислений. Классификация выражения длиной в 10 символов происходит за 5-7 секунд. Данная реализация обладает нестилизованным интерфейсом.

В таблице 1 представлено сравнение существующих решений.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Ощибка классификации, % | Распознава-ние выражения целиком | Вычисление значения | MacOS | Windows | Linux | Android | iOS | Web |
| MyScript Calculator | ≈99 | Да | Да | Да | Да | Да | Да | Да | Нет |
| Touch Calculator | ≈98 | Нет | Да | Да | Нет | Нет | Нет | Нет | Нет |
| Mathematical Expression Recognition | ≈79 | Да | Нет | Нет | Нет | Нет | Нет | Нет | Да |

Таблица 1

Существуют реализации для мобильных устройств, а также реализация для платформы MacOS. Самым востребованным решением является web-версия приложения: её можно использовать для любой из существующих операционных систем, как настольных, так и мобильных. У существующего web-приложения рукописного калькулятора есть реализация, однако у этой версии были выделены следующие недостатки: низкое качество классификации ( ≈79%) долгое распознавание выражения (10 символов за 5 секунд), не производит вычисление введенного выражения, а также не стилизованный интерфейс. В результате исследования существующих решений авторы предлагают рассмотреть альтернативную реализацию web-версии калькулятора с рукописным вводом, обеспечивающую распознавание математическое выражение с минимальной ошибкой классификации и его вычисление.

1. ***Математическая модель решения***

Создание приложения калькулятора с рукописным вводом состоит из следующих этапов:

* определение границ каждого из символов (сегментация изображения);
* определение класса каждого символа;
* посимвольный разбор получившейся строки и вычисление результата.

В качестве языка программирования для серверной части приложения был выбран язык Python, так как он предоставляет множество готовых библиотек, необходимых для работы с рукописным текстом, его классификацией и сегментацией.

Для определения границ символов и посимвольного разбора строки в языке Python существуют библиотеки, которые можно применить к серверной части приложения без изменений или предварительной обработки данных. Для определения класса каждого объекта существуют библиотеки с готовыми реализациями методов классификации, однако необходимо подбирать параметры для качественной работы методов.

Для выполнения классификации были рассмотрены следующие методы:

* мультиномиальная логистическая регрессия;
* наивный Байесовский классификатор;
* метод k ближайших соседей;
* метод Парзеновского окна;
* метод опорных векторов.

Рассмотрим подробнее принцип работы каждого из методов:

1. Мультиномиальная логистическая регрессия [4]

Модель мультиномиальной логистической регрессии предполагает, что на входе имеется множество объектов X, которому соответствует множество ответов Y. Имея на входе эти данные, предполагается 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.

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 h 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.

1. Наивный Байесовский классификатор [5, 6, 7]

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 Байесовский 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:

,

Где 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 сглаживания, 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:

1. Метод k ближайших соседей [8, 9]

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.

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.

1. Метод Парзеновского окна [9]

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.

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, воспользовавшись экспериментальным путем.

1. Метод опорных векторов [10, 11, 12, 13]

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 класс, который соответствует участку, в который попал классифицируемый объект.

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 <w,x>+b=0 для некоторых w и b, которая будет реализовывать разбиение выборки по классам.

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.: -1 < f(x) < 1. 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 λ=(λ1,…,λn) - 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 φ:Rn→X. 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 выборки, вводится набор дополнительных переменных εi ≥ 0, 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.

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

Эта задача так же, 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. Будем использовать Гауссову радиальную базисную функцию:

В Python существует библиотека sklearn[14], которая содержит в себе реализацию метода опорных векторов. Используем её для определения наилучших параметров при классификации выборки MNIST[15, 16].

1. ***Архитектура***

Для предложенной математической модели и поставленной задачи разработана архитектура, представленная на Рисунке 1.

1. Уровень представления

На данном уровне происходит графическое представление приложения пользователю

* 1. Компонент ввода выражения

Данный компонент считывает изображение выражения, введённого пользователем, и передает его контроллеру взаимодействия серверной части с пользователем.

* 1. Компонент отображения результата

Данный компонент отображает результат работы приложения на графическом интерфейсе и позволяет корректировать его. Он включает в себя:

* + 1. Модуль отображения результатов распознавания выражения и результата вычисления
    2. Компонент редактирования выражения

1. Уровень бизнес-логики

На данном уровне происходят все операции по доступному функционалу приложения, а также формируются запросы в базу данных.

* 1. Контроллер взаимодействия с пользователем

Позволяет осуществлять взаимодействие пользователя и серверной части

* 1. Компонент определения границ символом

Данный компонент принимает на вход изображение выражения, введенного пользователем, и возвращает изображения каждого символа по отдельности

* 1. Компонент классификации символов

Данный компонент принимает на вход набор символов, требующих классификации и возвращает набор классифицированных объектов

* 1. Компонент вычисления значения выражения

Данный компонент принимает на вход набор классифицированных символов в определенном порядке и возвращает значение выражения, которое создает последовательность символов на входе

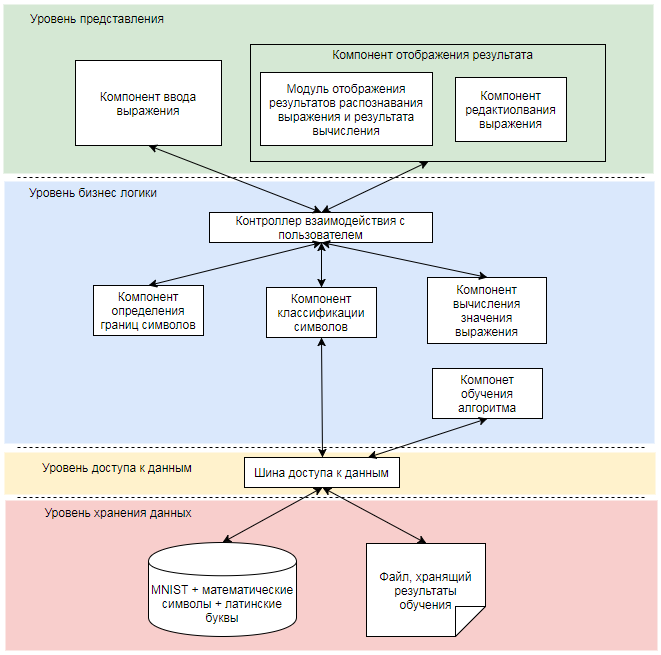
* 1. Компонент обучения алгоритма

Данный компонент используется только при необходимости переобучения выборки. Требуется только при первом запуске приложения, а также в случаях, если была изменена обучающая выборка.

1. Уровень доступа к данным
   1. Шина доступа к данным
2. Уровень хранения данных

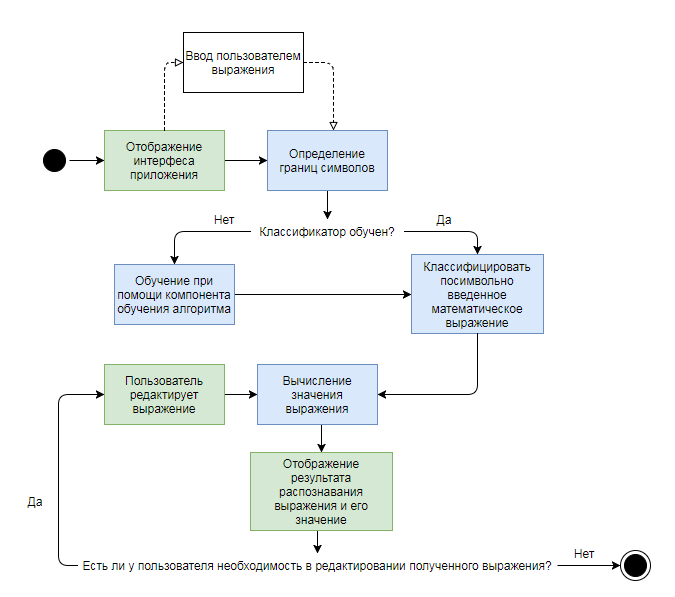
Содержит в себе выборку для обучения алгоритма классификации и результат последнего обучения

* 1. MNIST + Математические символы + латинские буквы
  2. Файл, хранящий результат обучения

****Рисунок 1

1. ***Сценарии использования***

Работа приложения, используя созданную архитектуру, приведена на Рисунке 2.

****Рисунок 2

Сценарии использования:

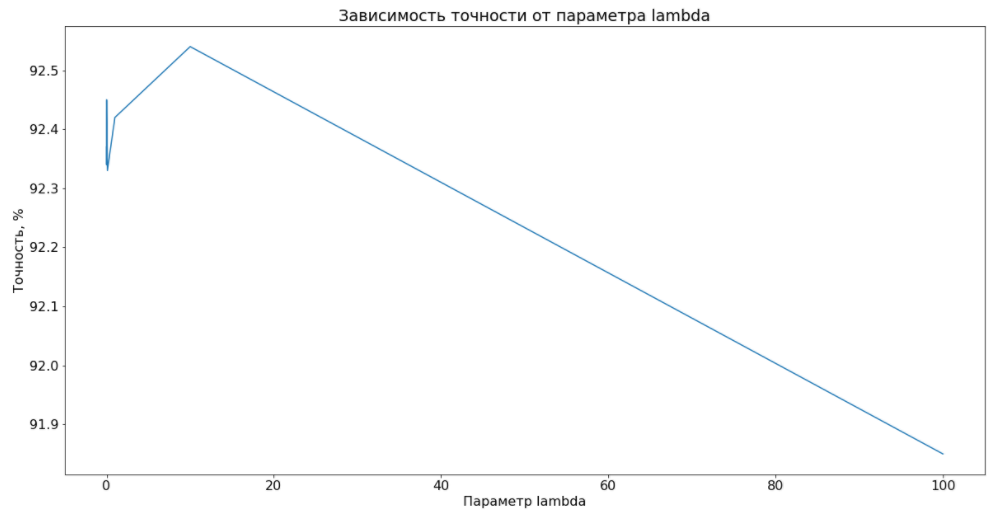
1. Распознавание выражения
   1. Пользователь заходит в приложение
   2. Пользователь видит поле для ввода выражения, кнопку «Вычислить значение выражения», кнопку «Очистить поле для ввода»
   3. Пользователь вводит математическое выражение в поле для ввода рукописного текста
   4. Пользователь нажимает кнопку «Вычислить значение выражения»
   5. Пользователь видит под кнопками текстовое поле с полученным после распознавания выражением и его результат
2. Редактирование результата распознавания
   1. Пользователь заходит в приложение
   2. Пользователь видит поле для ввода выражения, кнопку «Вычислить значение выражения», кнопку «Очистить поле для ввода»
   3. Пользователь вводит математическое выражение в поле для ввода рукописного текста
   4. Пользователь нажимает кнопку «Вычислить значение выражения»
   5. Пользователь видит под кнопками текстовое поле с полученным после распознавания выражением и его результат
   6. Пользователь получает некорректное значение
   7. Пользователь наводит курсор на распознанное выражение и ставит его в месте, где необходимо провести редактирование
   8. Пользователь редактирует выражение и автоматически видит новый ответ
3. Повторное распознавание выражения
   1. Пользователь находится в приложении
   2. Пользователь видит поле для ввода выражения с введённым на нём выражением, кнопку «Вычислить значение выражения», кнопку «Очистить поле для ввода», распознанное предыдущее выражение и его результат
   3. Пользовательно нажимает кнопку «Очистить поле для ввода» и видит чистое поле для ввода
   4. Пользователь вводит математическое выражение в поле для ввода рукописного текста
   5. Пользователь нажимает кнопку «Вычислить значение выражения»
   6. Пользователь видит под кнопками текстовое поле с полученным после распознавания выражением и его результат
4. ***Полученные результаты***

Перед нами стоит задача обучить классификатор таким образом, чтобы он осуществлял классификацию с максимальной точностью. Для этого необходимо произвести для каждого из методов подбор наилучших параметров.

* Мультиномиальная логистическая регрессия

Данный метод зависит от параметра 𝜆.

Результаты представлены на Рисунке 3

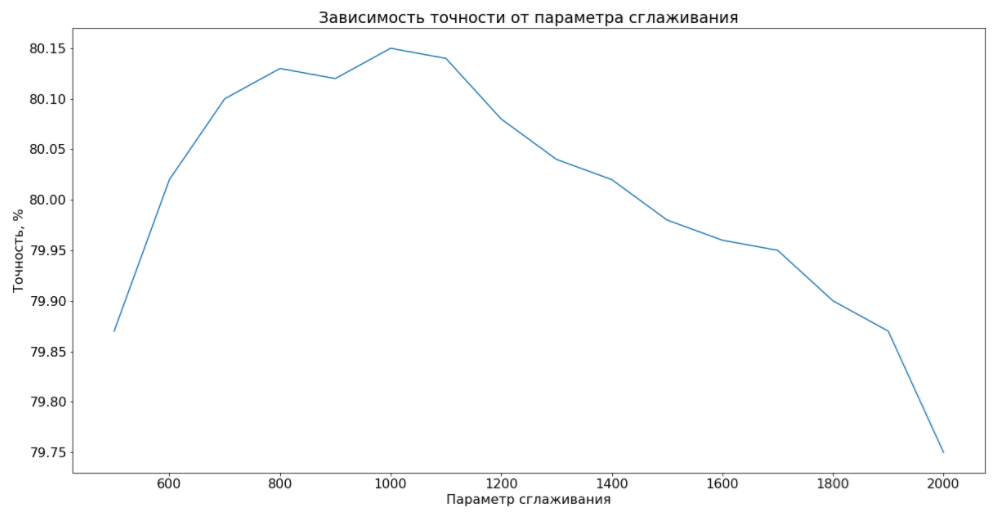
Рисунок 3

Наилучший результат – 92,54% достигается при значении параметра lambda = 0.01.

* Наивный Байесовский классификатор

Данный классификатор зависит от параметра сглаживания, благодаря которому значение функции классификатора не содержит 0 в знаменателе.

Результаты представлены на рисунке 4

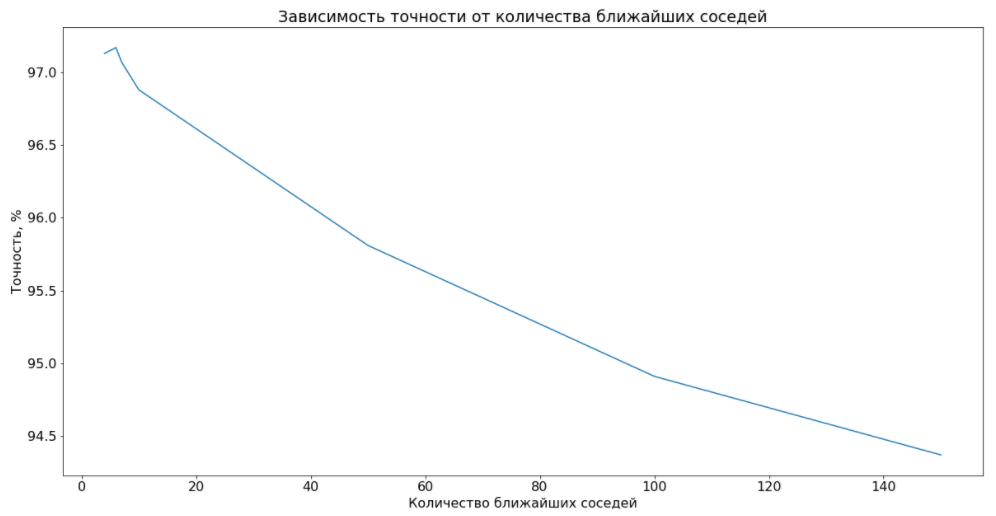
Рисунок 4

Наилучший результат – 80,15% достигнут при значении параметра сглаживания = 1000.

* Метод k ближайших соседей

Данный классификатор зависит от количества соседей, по которым происходит классификация объекта

Результаты представлены на рисунке 5

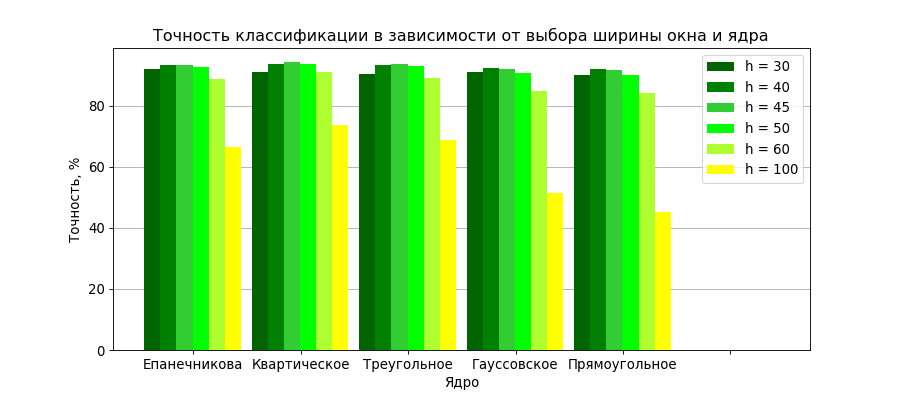
Рисунок 5

Наилучший результат – 97,17% достигнут при количестве ближайших соседей 6.

* Метод Парзеновского окна

Данный классификатор зависит от ширины выбранного окна и от выбранного ядра

Результаты представлены на рисунке 6

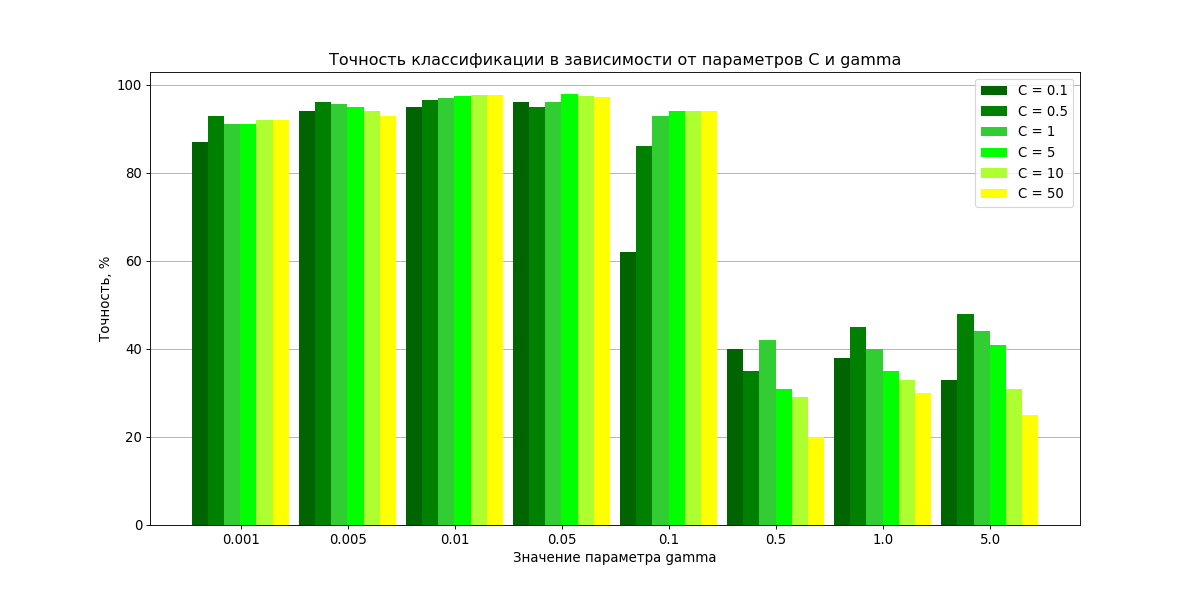
Рисунок 6

Наилучший результат – 94,14% достигнут при ширине окна h = 45 и Квартическом ядре.

* Метод опорных векторов с радиальным ядром функции (RBF)

Данный классификатор зависит от двух параметров: C и gamma.

Результаты представлены на рисунке 7

Рисунок 7

Наилучший результат – 98,52% достигнут при значении параметров C = 5 и gamma = 0.05.

Результаты, полученные после подбора наилучших параметров для выборки MNIST, приведены в Таблице 2:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Качество классификации (доля тестовой выборки равна 10%) | Скорость классификации 10000 объектов | Скорость обучения | Необходимость хранения выборки |
| Мультиномиальная логистическая регрессия | 92.54% | 0.04 секунды | 110 секунд | Нет |
| Наивный Байесовский классификатор | 92.44% | 0.02 секунды | 0,6 секунд | Нет |
| Метод k ближайших соседей | 97.15% | 10561 секунд | 110 секунд | Есть |
| Метод Парзеновского окна | 94.14% | 20082 секунды | 110 секунд | Есть |
| Метод опорных векторов с радиальным ядром функции (RBF) | 98.52% | секунд | 5426 секунд | Нет |

Таблица 2

Таким образом, наилучшее качество классификации на выборке MNIST показали методы k ближайших соседей и опорных векторов.

Метод k ближайших соседей требует хранения всей выборки, размеры которой приблизительно 80 Мб, а также для каждого нового классифицируемого объекта требуется переобучение классификатора, что сказывается на скорости работы приложения. Для одного символа классификация занимает примерно 1 секунду, то есть выражение в 5 символов займёт приблизительно 5 секунд. Для современного пользователя это очень долгое время ожидания результата.

Метод опорных векторов осуществляет классификацию символа всего за 0,001 секунды и не требует хранения данных, только значение полученных параметров при обучении. Данный подход обладает одним недостатком: долгое обучение, около 7 часов. Однако обучение требуется только один раз, поэтому после выполнения обучения на сервере пользователи смогут быстро получать результат.

Итак, наилучшее качество классификации на выборке MNIST показал метод опорных векторов, качество классификации составило 98,52%.

1. ***Дальнейшее развитие решения***

В первоначальной версии приложения предусмотрены только распознавание строковых выражений, без возведения в степень, извлечения корней, дробей и т.д. При дальнейшем развитии приложения можно добавить больше математических возможностей.

***Заключение***

Была изучена задача разработки калькулятора с рукописным вводом. Данная задача является актуальной и имеет широкий круг потенциальных пользователей. Рассмотрены существующие решения и выявлены их недостатки. Но основе исследования существующих реализаций приложения калькулятора с рукописным вводом для реализации приложения выбрана web-платформа. Изучены методы классификации рукописного текста. Проведены тесты по подбору наилучших параметров для каждого из методов на основе базы данных рукописных цифр MNIST. Наилучшие результаты показал метод опорных векторов с ядром RBF и значениями параметров C = 5 и gamma = 0.05. Точность классификации составила 98,52%, время обучения – приблизительно 7 часов, однако обучение требуется только при первом запуске приложения. Разработаны сценарии использования приложения и на их основе предложена архитектура будущего приложения.

***Литература***

1. <https://www.youtube.com/watch?v=gm4-3LACUfA> (дата обращения: 25.01.2018)
2. <https://www.youtube.com/watch?v=9xGWnnozi-M> (дата обращения: 25.01.2018)
3. <http://cat.prhlt.upv.es/mer/> (дата обращения: 25.01.2018)
4. <https://en.wikipedia.org/wiki/Multinomial_logistic_regression> (дата обращения 10.09.2017)
5. [http://bazhenov.me/blog/2012/06/11/naive-bayes.html](http://bazhenov.me/blog/2012/06/11/naive-bayes.html%20) (дата обращения: 17.09.2017)
6. [https://habrahabr.ru/post/120194/](https://habrahabr.ru/post/120194/%20) (дата обращения: 17.09.2017)
7. [http://www.machinelearning.ru/wiki/index.php?title=Наивный\_байесовский\_классификатор](http://www.machinelearning.ru/wiki/index.php?title=Наивный_байесовский_классификатор%20) (дата обращения: 17.09.2017)
8. [http://www.machinelearning.ru/wiki/index.php?title=Метод\_ближайшего\_соседа](http://www.machinelearning.ru/wiki/index.php?title=Метод_ближайшего_соседа%20) (дата обращения: 21.09.2017)
9. Курс лекций в Санкт-Петербургском политехническом университете Петра Великого, 2002 г. Уткин Л.В Машинное обучение (Machine Learning) Метрические методы классификации и регрессии, с. 4 - 57
10. [http://www.machinelearning.ru/wiki/index.php?title=SVM](http://www.machinelearning.ru/wiki/index.php?title=SVM%20) (дата обращения: 02.10.2017)
11. Курс лекций по машинному обучению, 21.12.2007 г., К.В. Воронцов “Лекции по методу опорных векторов”
12. Alexey Nefedov ,Support Vector Machines: A Simple Tutorial, 2016 г., с. 2-34
13. Курс «Алгоритмы для Интернета», 9.11.2006, Юрий Лифшиц, Алгоритмы для интернета. Метод опорных векторов, с.1-9
14. <http://scikit-learn.org/stable/> (дата обращения: 28.01.2018)
15. <https://ru.wikipedia.org/wiki/MNIST_(база_данных)> (дата обращения: 28.01.2018)
16. <http://yann.lecun.com/exdb/mnist/> (дата обращения: 28.01.2018)