УДК 004.89

**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, Россия)*

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, распознавание текста.*

***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.

У всех пользователей различные привычки и уникальный опыт: одним людям удобен голосовой ввод, другие привыкли к клавишам калькулятора, но есть и те, кому удобен рукописный ввод. В статье предлагается рассмотреть возможности создания калькулятора с рукописным вводом.

**Существующие решения**

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

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

1. ***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.

1. ***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.

1. ***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 определения результата вычислений. Классификация выражения длиной в 10 символов происходит за 5-7 секунд. Кроме того, данная реализация обладает нестилизованным интерфейсом.

В таблице 1 представлено наличие реализаций существующих решений для различных платформ.

В таблице 2 представлен сравнительный анализ функционала существующих решений.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Таблица 1. Наличие реализаций существующих решений для различных платформ* | | | | | | |
| **Решение** | **MacOS** | **Windows** | **Linux** | **Android** | **iOS** | **Web** |
| MyScript Calculator | Нет | Нет | Нет | Да | Да | Нет |
| Touch-Calculator | Да | Нет | Нет | Нет | Нет | Нет |
| Mathematical Expression Recognition | Нет | Нет | Нет | Нет | Нет | Да |
| *Таблица 2. Сравнительный анализ функционала существующих решений* | | | | | | |
| **Решение** | **Точность классификации, %** | | **Распознавание выражения целиком** | | **Вычисление значения** | |
| MyScript Calculator | ≈ 99 | | Да | | Да | |
| Touch-Calculator | ≈ 98 | | Нет | | Да | |
| Mathematical Expression Recognition | ≈ 79 | | Да | | Нет | |

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 стилизован интерфейс. В результате исследования существующих решений предлагается рассмотреть альтернативную реализацию web-версии калькулятора с рукописным вводом, обеспечивающую распознавание математического выражение с минимальной ошибкой классификации и его вычисление.

**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;
* обучение классификатора.

В качестве языка программирования для серверной части приложения был выбран язык Python, так как он предоставляет множество готовых библиотек 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 применимость перечисленных методов.

1. ***Мультиномиальная логистическая регрессия [4].***

Модель мультиномиальной логистической регрессии предполагает гипотезу, 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.

Далее определяются параметры модели. Подбор параметров в данной модели осуществляется при помощи минимизации функции стоимости, которая описывается следующим образом:

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

Для минимизации функции рассчитывается градиент. Затем он используется для минимизации функции стоимости.

В результате 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 модели Байесовского классификатора 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*.

Байесовский классификатор осуществляет классификацию посредством оценки априорного максимума:

Вероятность *P(x)* для всех объектов одинакова, значит формулу можно записать в следующем виде:

Для вычисления класса объекта рассчитывается вероятность, с которой объект принадлежит классу *k*.

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

Наивный Байесовский классификатор приобретает вид:

При большом количестве признаков происходит многократное перемножение чисел меньше единицы, поэтому используется формула в логарифмическом пространстве:

Безусловная вероятность того, что объект принадлежит классу *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.

Конечный вид наивного Байесовского классификатора имеет следующий вид:

Используя данную формулу, определяется класс объекта. Параметр *a* подбирается экспериментальным путём.

1. ***Метод k ближайших соседей [8, 9].***

В модели метода *k* ближайших соседей выбирается количество ближайших соседей *k*, по которым будет происходить оценка классифицируемого объекта.

Значение *k* определяется по критерию скользящего контроля [10].

Все объекты тренировочной выборки располагаются в следующей последовательности:

,

где - функция расстояния.

Из полученной последовательности выбирается *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.

Функция для классификации объекта выглядит следующим образом:

где *wix*- вес *i*-го объекта из упорядоченной по расстоянию тренировочной выборки для объекта *x*.

Используя полученную функцию и подобранное количество ближайших соседей, определяется класс объекта.

1. ***Метод Парзеновского окна [9].***

В основе модели метода Парзеновского окна лежит модель метода *k* ближайших соседей. В методе ближайших соседей выбирается количество ближайших соседей *k*, по которым будет происходить оценка классифицируемого объекта. В данном алгоритме по подобной логике выбирается ширина Парзеновского окна.

Как и для параметра *k* из метода ближайших соседей, значение *h* определяется по критерию скользящего контроля [10].

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

,

где - функция расстояния.

Функция для классификации объекта выглядит следующим образом:

где *K* – ядро.

Ядро может быть выбрано из следующего набора ядер:

* Ядро Епанечникова;
* квартическое ядро;
* треугольное ядро;
* Гауссовское ядро;
* Прямоугольное ядро.

Функция ядра выбирается экспериментальным путем.

Используя полученную функцию, выбранные ширину окна и вид ядра, определяется класс объекта.

1. ***Метод опорных векторов [11, 12, 13, 14].***

Модель метода опорных векторов была разработана для бинарной классификации, однако существует и модификация для многоклассовой. Все объекты тренировочной выборки представлены в *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*. Таких плоскостей можно провести бесконечно большое количество. В алгоритме опорных векторов лучшей разделяющей плоскостью считается плоскость, расстояние от которой до каждого из классов максимально. Пространство оказывается разделено на участки, каждый из которых соответствует какому-либо классу.

После того, как плоскость проведена, определяется положение каждого классифицируемого объекта. Ему присваивается класс, который соответствует участку, в который попал классифицируемый объект.

Пусть *X* – пространство объектов тренировочной выборки, Y – пространство ответов. В случае, когда необходимо разделить линейно неразделимую выборку, все элементы этой выборки вкладываются в пространство, размерность которого выше, чем размерность заданной выборки, используется отображение *φ:R(n)→X*. Это отображение выбирается таким образом, чтобы в новом пространстве выборка была линейно-разделимой. *X* – является пространством со скалярным произведением. Разделяющая функция в данном случае имеет следующий вид:

где коэффициенты 𝜆i зависят от yi и от значения ядра. Ядром является любая функция вида K(x,y), являющаяся симметричной и неотрицательно определенной.

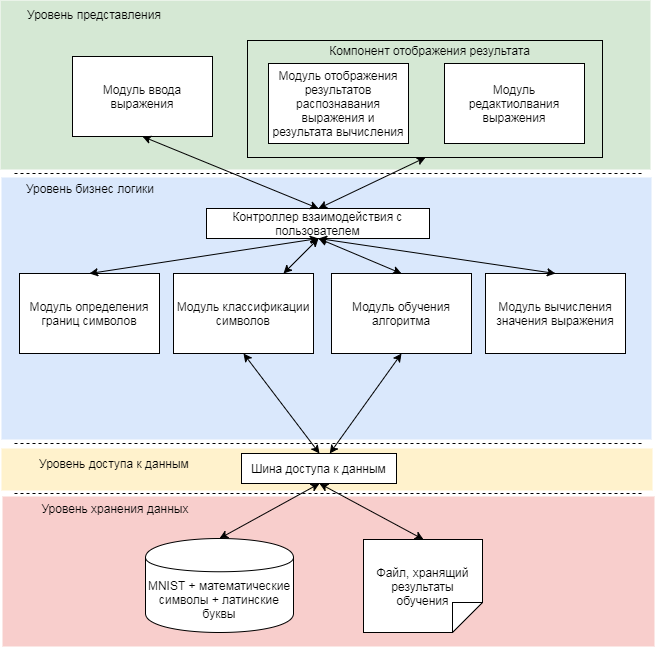
В данной статье рассматривается ядро вида:

*, γ>0.*

Такое ядро называется Радиальной базисной функцией (RBF).

**Архитектура**

Для решения задачи разработки калькулятора с рукописным вводом разработана архитектура, представленная на рисунке 1.



*Рис. 1. Архитектура приложения калькулятора с рукописным вводом*

***1. Уровень представления***

На уровне представления считывается введенное пользователем изображение, отправляется запрос к серверной части приложения для распознавания или редактирования текста, отображается результат распознавания выражения и его результат.

Модуль ввода выражения позволяет пользователю изобразить математическое выражение, считать его и передать его серверной части приложения

Компонент отображения результата отвечает за работу модуля отображения результата распознавания выражения и результата вычислений, а также модуля редактирования выражения.

Модуль отображения результата распознавания выражения и результата вычислений позволяет отобразить на интерфейсе результат распознавания и классификации рукописного текста, а также его результат. Модуль производит взаимодействие с серверной частью приложения, отвечающей за классификацию рукописных символов и их редактирование.

Модуль редактирования выражения позволяет пользователю редактировать некорректно распознанные символы рукописного математического выражения, передает исправленный вариант серверной части приложения, отвечающей за вычисление результата.

***2. Уровень бизнес-логики***

На уровне бизнес-логики происходит принятие запроса от пользователя на распознавание изображения или на редактирование выражения, сегментация полученного изображения, классификация символов, вычисление результата выражения, обучение классификатора, формирование ответа пользователю с распознанным математическим выражением и результатом его вычисления. Также на данном уровне формируются запросы на получение данных из базы данных.

Контроллер взаимодействия с пользователем принимает запросы от пользователей, перенаправляет их на конкретные модули серверной части приложения, отправляет ответы сервера на запросы клиентов.

Модуль определения границ и символов отвечает за сегментацию полученного выражения, определяет границы символов и отправляет результат на контроллер.

Модуль классификации символов принимает на вход изображения отдельных символов, запрашивает данные из файла, в котором содержатся результаты обучения классификатора, при помощи полученных параметров производит классификацию каждого из полученных изображений, отправляет результат контроллеру.

Модуль вычисления значения выражения принимает на вход набор символов в определенном порядке, распознает правильный порядок вычислений и производит их, затем возвращает результат на контроллер.

Модуль обучения алгоритма используется только в тех случаях, когда база данных символов для обучения была изменена или дополнена. Модуль принимает запрос на обучение; формирует запрос в базу данных символов; получает данные из базы данных; используя полученные данные, производит обучение алгоритма; записывает результат в файл; отправляет уведомление контроллеру о том, что обучение окончено.

***3. Уровень доступа к данным***

Является контроллером запросов к базе данных и ответов на них.

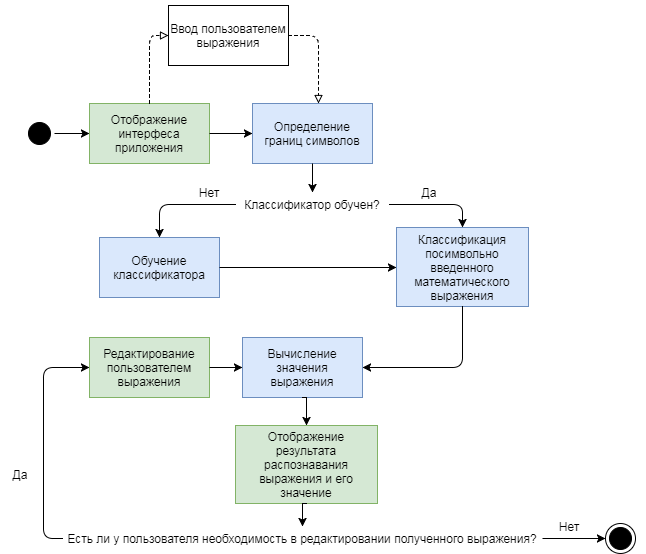
***4. Уровень хранения данных***

Содержит базы данных для обучения классификатора и сериализованный объект обученного классификатора.

Рассмотрим возможные сценарии использования приложения.

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 на место, где необходимо произвести 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. Пользователь 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

Работа приложения, используя созданную архитектуру и предложенные сценарии, приведена на рисунке 2.



*Рис. 2. Работа приложения, используя предложенную архитектуру и сценарии использования*

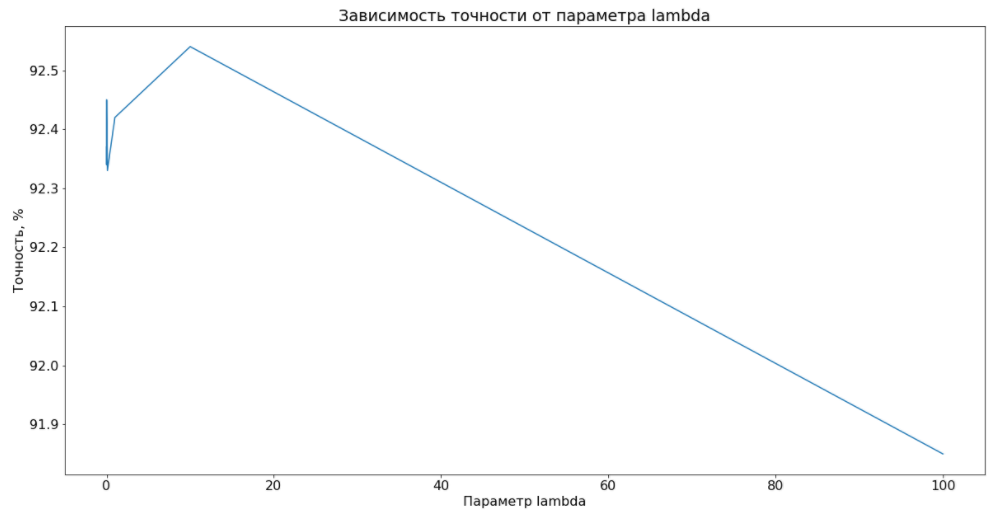
**Полученные результаты**

Классификатор нужно обучить таким образом, чтобы он осуществлял классификацию с максимальной точностью. Исследуем каждый из предложенных методов классификации.

1. ***Мультиномиальная логистическая регрессия.***

Данный метод зависит от параметра 𝜆, который используется в функции стоимости.

Результаты исследования зависимости точности классификации от выбора параметра 𝜆 представлены на рисунке 3.

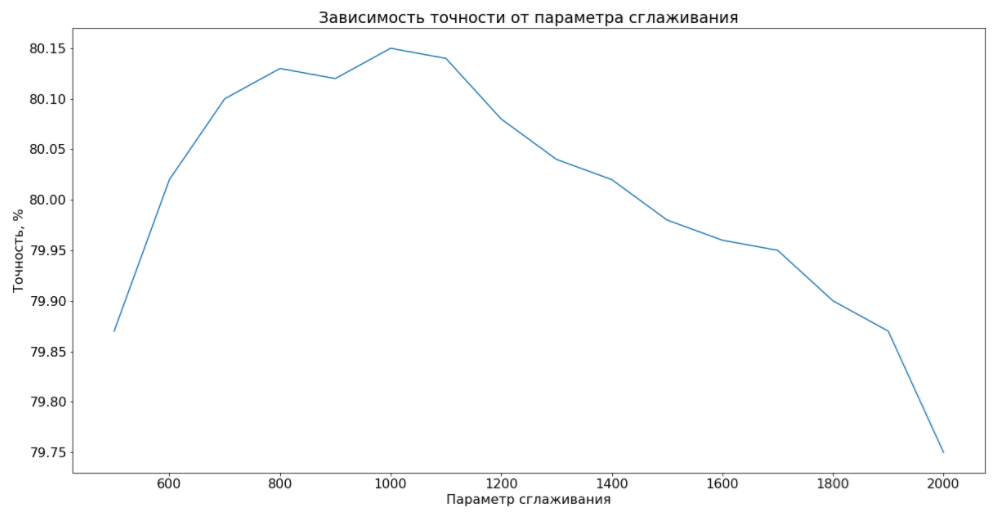
*Рис. 3. Зависимость точности классификации методом мультиномиальной регрессии от выбора параметра 𝜆*

Достигнута точность 92,54% при значении параметра 𝜆 = 0.01.

1. ***Наивный Байесовский классификатор.***

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

Результаты исследования зависимости точности классификации от параметра сглаживания представлены на рисунке 4.

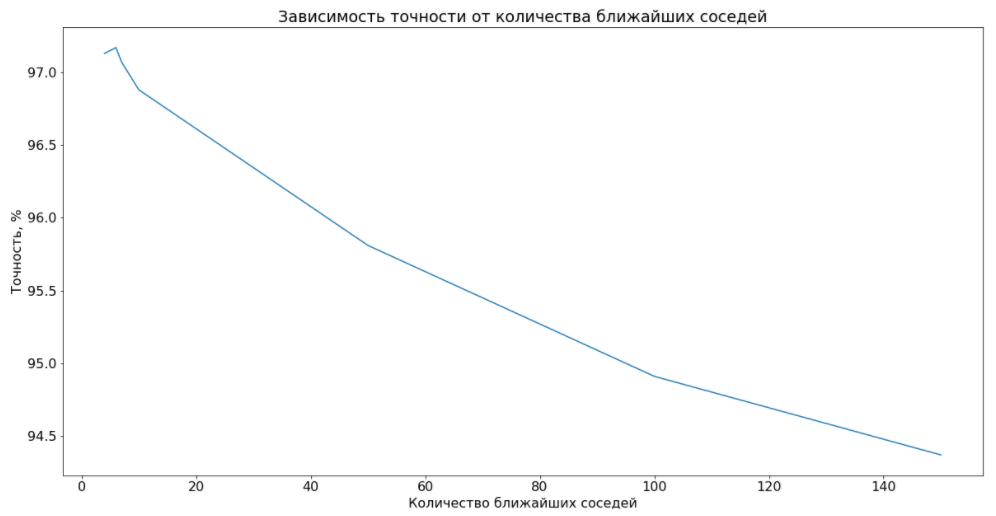
*Рис. 4. Зависимость точности классификации методом наивного Байесовского классификатора от параметра сглаживания*

Достигнута точность 80,15% при значении параметра сглаживания равном 1000.

1. ***Метод k ближайших соседей.***

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

Результаты исследования зависимости точности классификации от количества ближайших соседей представлены на рисунке 5.

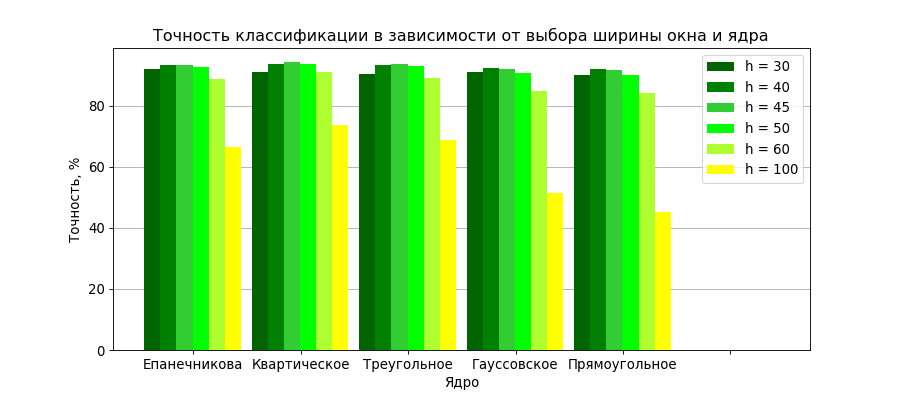
*Рис. 5. Зависимость точности классификации методом k ближайших соседей от количества ближайших соседей*

Достигнута точность 97,17% при количестве ближайших соседей равном 6.

1. ***Метод Парзеновского окна.***

Данный классификатор зависит от ширины выбранного окна и от выбранного ядра.

Результаты исследования зависимости точности классификации от ширины окна и от вида ядра представлены на рисунке 6.

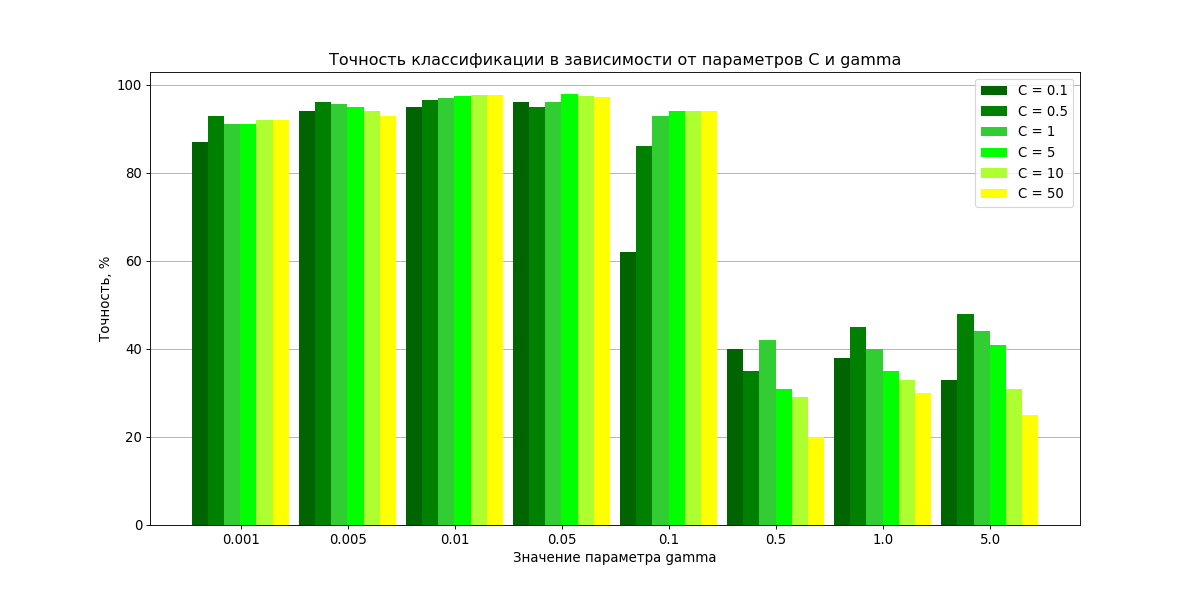
*Рис. 6. Зависимость точности классификации методом Парзеновского окна от ширины окна и вида ядра*

Достигнута точность 94,14% при ширине окна h = 45, в качестве ядра было выбрано Квартическое ядро.

1. ***Метод опорных векторов с радиальным ядром функции (RBF)***

Данный классификатор зависит от двух параметров: C и γ.

Результаты исследования зависимости точности классификации от параметров C и γ представлены на рисунке 7.

*Рис. 7. График зависимости точности классификации методом опорных векторов от выбранных параметров C и γ*

Достигнута точность 98,52% при значении параметров C = 5 и γ= 0.05.

Подведем итоги исследования методов классификации. Результаты, полученные после подбора наилучших параметров для выборки MNIST, приведены в таблице 3:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Таблица 3. Исследование методов классификации* | | | | |
|  | **Качество классификации (доля тестовой выборки равна 10%), %** | **Время классификации 10000 объектов, секунды** | **Время обучения, секунды** | **Переобучение классификатора для каждого символа** |
| Мультиномиальная логистическая регрессия | 92.54 | 0.04 | 110 | Нет |
| Наивный Байесовский классификатор | 92.44 | 0.02 | 0,6 | Нет |
| Метод k ближайших соседей | 97.15 | 10561 | 110 | Да |
| Метод Парзеновского окна | 94.14 | 20082 | 110 | Да |
| Метод опорных векторов с радиальным ядром функции (RBF) | 98.52 | 256 | 5426 | Нет |

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 время классификации выражения в 5 символов составит приблизительно 5 секунд.

Метод опорных векторов осуществляет классификацию символа приблизительно за 0,026 секунд и не требует хранения данных, только значение полученных параметров при обучении. Данный подход обладает одним недостатком: обучение длится около полутора часов. Однако обучение требуется только один раз, поэтому после выполнения обучения на сервере пользователи смогут получить результат распознавания выражения в 10 символов приблизительно за 0,2 секунды.

***Заключение***

Для решения задачи создания калькулятора с рукописным вводом в качестве классификатора выбран метод опорных векторов, так как при сравнительном анализе с другими методами он классифицировал объекты базы данных MNIST с максимальной среди исследованных методов точностью (98,52%) и высокой скоростью классификации (приблизительно 10 символов за 0,2 секунды).

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.

***Литература:***

1. MyScript Calculator // YouTube 2014 URL: <https://www.youtube.com/watch?v=gm4-3LACUfA> (дата обращения: 25.01.2018)
2. Touch Calculator // YouTube 2015 URL: <https://www.youtube.com/watch?v=9xGWnnozi-M> (дата обращения: 25.01.2018)
3. Mathematical Expression Recognition // 2018 URL: <http://cat.prhlt.upv.es/mer/> (дата обращения: 25.01.2018)
4. Multinomial logistic regression // Wikipedia 2017 URL: <https://en.wikipedia.org/wiki/Multinomial_logistic_regression> (дата обращения 10.09.2017)
5. Наивный Байесовский классификатор // 2012 URL: [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. Наивный Байесовский классификатор в 25 строк кода // Хабрахабр 2011 URL: [https://habrahabr.ru/post/120194/](https://habrahabr.ru/post/120194/%20) (дата обращения: 17.09.2017)
7. Наивный Байесовский классификатор // Распознавание 2010 URL: [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. Метод ближайших соседей // Распознавание 2015 URL: [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. Уткин Л.В. Курс лекций в Санкт-Петербургском политехническом университете Петра Великого, 2002 г. Машинное обучение (Machine Learning). Метрические методы классификации и регрессии, с. 4 – 57
10. Скользящий контроль // Распознавание 2010 URL: <http://www.machinelearning.ru/wiki/index.php?title=Кросс-валидация> (дата обращения: 05.02.2018)
11. Машина опорных векторов // Распознавание 2016 URL: [http://www.machinelearning.ru/wiki/index.php?title=SVM](http://www.machinelearning.ru/wiki/index.php?title=SVM%20) (дата обращения: 02.10.2017)
12. Воронцов К.В. Курс лекций по машинному обучению, 21.12.2007 г. “Лекции по методу опорных векторов”
13. Nefedov Alexey. Support Vector Machines: A Simple Tutorial, 2016 г., с. 2-34
14. Лифшиц Юрий. Курс «Алгоритмы для Интернета», 09.11.2006. Алгоритмы для интернета. Метод опорных векторов, с.1-9
15. Scikit learn library // 2017 URL: <http://scikit-learn.org/stable/> (дата обращения: 28.01.2018)
16. MNIST(база данных) // Wikipedia 2017 URL: <https://ru.wikipedia.org/wiki/MNIST_(база_данных)> (дата обращения: 28.01.2018)
17. The MNIST database // URL: <http://yann.lecun.com/exdb/mnist/> (дата обращения: 28.01.2018)