УДК 004.89
DOI: 10.15827/2311-6749.18.1.6

# *ПОДХОДЫ К СОЗДАНИЮ КАЛЬКУЛЯТОРА С РУКОПИСНЫМ ВВОДОМ*

*С.А. Беляев, к.т.н., доцент, bserge@bk.ru; Д.А. Лапцевич, студентка, darya.laptsevich@gmail.com*
*(Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина) (СПбГЭТУ «ЛЭТИ»), ул. Профессора Попова, 5, г. 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 предложены 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 с использованием образцов рукописного написания цифр MNIST.

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

Сегодня техника окружает нас всюду, более того, ее интеллектуальные способности развиваются год от года, поэтому для разработчиков ПО важно добиться удобства и практичности в «общении» пользователя с техническими средствами.

В числе самых популярных инструментов человеко-машинного интерфейса – голосовой ввод и ввод рукописного текста. Эти возможности стали нормой и функциональным инструментом для современного пользователя.

Практически каждый человек использует калькулятор. Любая сфера деятельности предполагает какие-либо расчеты: основной задачей бухгалтера является формирование баланса организации, домохозяйка подсчитывает стоимость коммунальных услуг, студент вычисляет ответ математической задачи. Каждый из нас сталкивается с вычислениями, которые в некоторых случаях затруднительно выполнять без вычислительной машины.

У людей различные привычки и уникальный опыт: одним удобен голосовой ввод, другие привыкли к клавишам калькулятора, а кому-то проще пользоваться рукописным вводом. В статье предлагается рассмотреть возможности по созданию калькулятора с рукописным вводом.

## *Существующие решения*

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

– MyScript Calculator [1];
– Touch-Calculator [2];
– Mathematical Expression Recognition [3].

Практически универсальным на сегодняшний день является MyScript Calculator: 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.

Реализация Touch-Calculator с рукописным вводом осуществляется 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 – распознавание происходит только по одному символу.

Классификация символов математического выражения в Mathematical Expression Recognition происходит некачественно (из 100 введенных символов 21 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. Существующая реализация обладает нестилизованным интерфейсом.

В таблице 1 показано 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: ее можно использовать для любой из существующих операционных систем, как настольных, так и мобильных. У существующего web-приложения рукописного калькулятора можно выделить несколько недостатков: низкое качество классификации (≈79 %) долгое распознавание выражения (10 символов за 5 секунд) и он не производит вычисление введенного выражения. 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***

Решение задачи рукописного ввода для калькулятора осуществляется в несколько 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 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 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 рассмотрим следующие методы:

– мультиномиальная логистическая регрессия;

– наивный Байесовский классификатор;

– метод *k* ближайших соседей;

– метод Парзеновского окна;

– метод опорных векторов.

**Мультиномиальная логистическая регрессия**

Модель мультиномиальной логистической регрессии [4] предлагает гипотезу, которая оценивает вероятность $P = (y = k|x)$, где $k = 1, \ldots, K$, 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:

$$J(w) = -\left[\sum_{i=1}^{N}\sum_{k=1}^{K} 1\{y_i = k\}\left(\log\left(e^{\left(w_k^T x_i\right)}\right) - log\left(\sum_{j=1}^{K} e^{w_j^T x_i}\right)\right)\right].$$

При использовании мультиномиальной регрессии вероятность попадания объекта в класс определяется по формуле

$$P(y_i = k \mid x_i; w) = \frac{e^{w_k{}^T x_i}}{\sum_{j=1}^{K} e^{w_j{}^T x_i}}.$$

Минимизация функции стоимости осуществляется с использованием градиента.

В результате получаем параметры математической модели, которые будут использованы для подтверждения гипотезы. С ее применением по параметрам входного тестового объекта определяется его класс.

**Наивный Байесовский классификатор**

В основе модели Байесовского классификатора [5, 6] лежит теорема Байеса:

$$P(k \mid x) = \frac{P(x \mid k)P(k)}{P(x)},$$

где $P(k|x)$ – вероятность того, что объект $x$ принадлежит классу $k$; $P(x|k)$ – вероятность того, что объект $x$ встречается среди объектов класса $k$; $P(k)$ – безусловная вероятность встретить объект класса $k$, $P(x)$ – безусловная вероятность объекта $x$.

Байесовский классификатор осуществляет классификацию с использованием оценки априорного максимума:

$$k_{max} = arg\,max_{k \in K} \frac{P(x \mid k)P(k)}{P(x)}.$$

Вероятность $P(x)$ для всех объектов одинакова, формула может быть записана в следующем виде:

$$k_{max} = arg\,max_{k \in K} P(x|k)P(k).$$

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:

$$P(x \mid k) = P(w_1 \mid k)P(w_2 \mid k)...P(w_N \mid k) = \prod_{i=1}^{N} P(w_i \mid k).$$

Наивный Байесовский классификатор приобретает вид:

$$k_{max} = arg\,max_{k \in K} P(k) \prod_{i=1}^{N} P(w_i|k).$$

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

$$k_{max} = arg\,max_{k \in K} (log\,P(k) + \sum_{i=1}^{N} \log P(w_i|k).$$

Безусловная вероятность того, что объект принадлежит классу $k$, оценивается по тренировочной выборке:

$$P(k) = N_k/N,$$

где $N_k$ – количество объектов в тренировочной выборке, принадлежащих классу $k$; $N$ – количество всех объектов в тренировочной выборке.

Оценка параметров Байесовской модели:

$$P(w_i \mid k) = \frac{w_{ik} + a}{a + \sum_{j=1}^{N} w_{jk}},$$

где $w_{ik}$ – общее количество элементов с заданным значением признака $i$ в классе $k$; $a$ – параметр сглаживания, значение которого всегда больше 0, вводится, чтобы значение вероятности не принимало нулевое значение.

Окончательный вариант наивного Байесовского классификатора принимает следующий вид:

$$k_{max} = arg\,max_{k \in K}\, log \frac{N_k}{N} \sum_{i=1}^{M} log \frac{w_{ik} + a}{a + \sum_{j=1}^{M} w_{jk}}.$$

Используя данную формулу, определяется класс объекта. Параметр $a$ подбирается экспериментальным путем.

**Метод *k* ближайших соседей**

В модели метода $k$ ближайших соседей [7] выбирается количество ближайших соседей $k$, по которым будет происходить оценка классифицируемого объекта.

Значение *k* определяется по критерию скользящего контроля [8].

Все объекты тренировочной выборки располагаются в следующей последовательности:

$$\rho(x, x_{i1}) \le \rho(x, x_{i2}) \le ... \le \rho(x, x_{iN}),$$

где $\rho(x, x_{in})$ – функция расстояния.

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

Функция для классификации объекта выглядит следующим образом:

$$f(x) = arg\,max_{y \in Y} \sum_{i=1}^{k} \left(y(x_{xi}) = y\right) w_{xi},$$

где $w_{ix}$– вес *i*-го объекта из упорядоченной по расстоянию тренировочной выборки для объекта *x*.

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

### Метод Парзеновского окна

В основе модели метода Парзеновского окна [9] 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 [8].

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

$$\rho(x, x_{i1}) \le \rho(x, x_{i2}) \le ... \le \rho(x, x_{iN}),$$

где $\rho(x, x_{in})$ – функция расстояния.

Функция для классификации объекта выглядит следующим образом:

$$f(x) = arg\,max_{y \in Y} \sum_{i=1}^{n} \left(y(x_{xi}) = y\right) K\left(\frac{1}{h}\rho(z, x_{xi})\right),$$

где *K* – ядро.

Ядро может быть выбрано из следующего набора:

– ядро Епанечникова;

– квартическое;

– треугольное;

– гауссовское;

– прямоугольное.

Функция ядра подбирается экспериментальным путем. Используя полученную функцию и выбранную ширину окна, определяется класс объекта.

### Метод опорных векторов

Модель метода опорных векторов [8] 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 разделенным 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 помещаются в пространство, размерность которого выше размерности заданной выборки, используется отображение $\varphi: R^{(n)} \rightarrow X$. 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:

$$f(x) = \left(w, \varphi(x_i)\right) + b, w = \sum_{i=1}^{N} \lambda_i y_i \varphi(x_i),$$

где коэффициенты $\lambda_i$ зависят от $y_i$ и от значения ядра. Ядром является любая функция вида $K(x, y)$, являющаяся симметричной и неотрицательно определенной.

В данной статье рассматривается ядро вида:

$K(x, y) = e^{-\gamma\|x-y\|^2}$, $\gamma > 0$.

Такое ядро называется радиальной базисной функцией (RBF).

***Архитектура***

Для разработки калькулятора с рукописным вводом предлагается архитектура, представленная на рисунке 1.

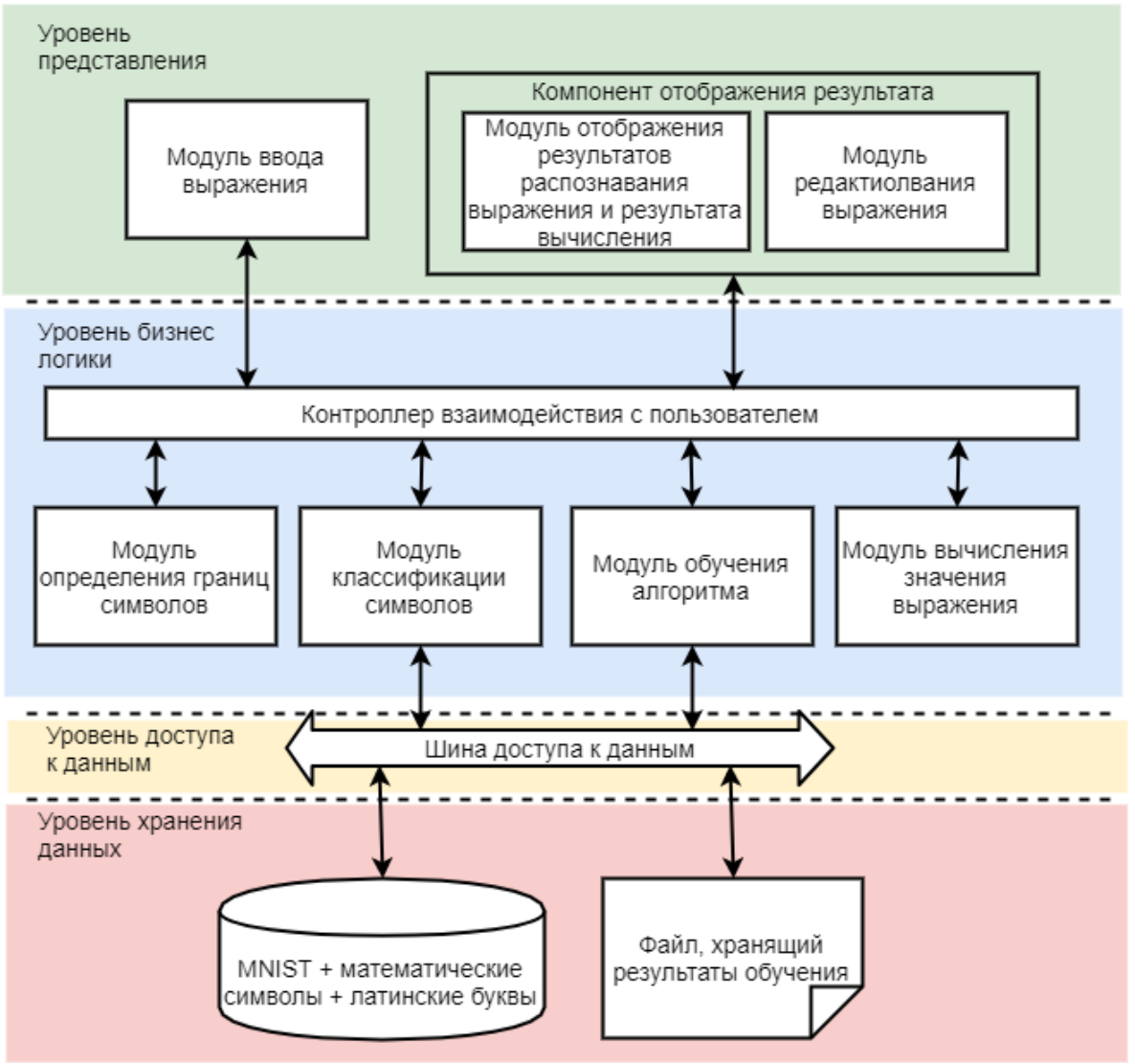


*Рис. 1. Архитектура приложения калькулятора с рукописным вводом*

*Уровень представления*

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

Модуль ввода выражения позволяет пользователю написать 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 резуль-

тат. 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.

*Уровень бизнес-логики*

На уровне бизнес-логики происходят 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 БД.

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 БД символов для обучения была изменена или дополнена. Модуль принимает запрос на обучение; формирует запрос в БД символов; получает данные из БД; используя полученные данные, обеспечивает обучение алгоритма; записывает результат в файл; отправляет уведомление контроллеру о том, что обучение окончено.

*Уровень доступа к данным*

На данном уровне обеспечивается работа шины работы с данными, которая является контроллером запросов к БД и ответов на них.

*Уровень хранения данных*

Содержит БД для обучения классификатора и сериализованный объект обученного классификатора.

***Сценарии использования***

Рассмотрим возможные сценарии использования приложения.

1. Распознавание пользователем выражения:

– заходит в приложение;

– видит поле для ввода выражения, кнопку «Вычислить значение выражения», кнопку «Очистить поле для ввода»;

– вводит математическое выражение в поле для ввода рукописного текста;

– нажимает кнопку «Вычислить значение выражения»;

– видит под кнопками текстовое поле с полученным после распознавания выражением и его результат.

2. Редактирование пользователем результата распознавания:

– заходит в приложение;

– видит поле для ввода выражения, кнопку «Вычислить значение выражения», кнопку «Очистить поле для ввода»;

– вводит математическое выражение в поле для ввода рукописного текста;

– нажимает кнопку «Вычислить значение выражения»;

– видит под кнопками текстовое поле с полученным после распознавания выражением и его результат;

– получает некорректное значение;

– наводит курсор на распознанное выражение и ставит его на место, где необходимо произвести редактирование;

– редактирует выражение и автоматически видит новый ответ.

3. Повторное распознавание выражения:

– пользователь находится в приложении;

– видит поле для ввода выражения с введенным на нем выражением, кнопку «Вычислить значение выражения», кнопку «Очистить поле для ввода», распознанное предыдущее выражение и его результат;

– нажимает кнопку «Очистить поле для ввода» и видит чистое поле для ввода;

– вводит математическое выражение в поле для ввода рукописного текста;

– нажимает кнопку «Вычислить значение выражения»;

– видит под кнопками текстовое поле с полученным после распознавания выражением и его результат.

Схема работы приложения, построенного на основании предложенной архитектуры и описанных сценариев использования, приведена на рисунке 2.

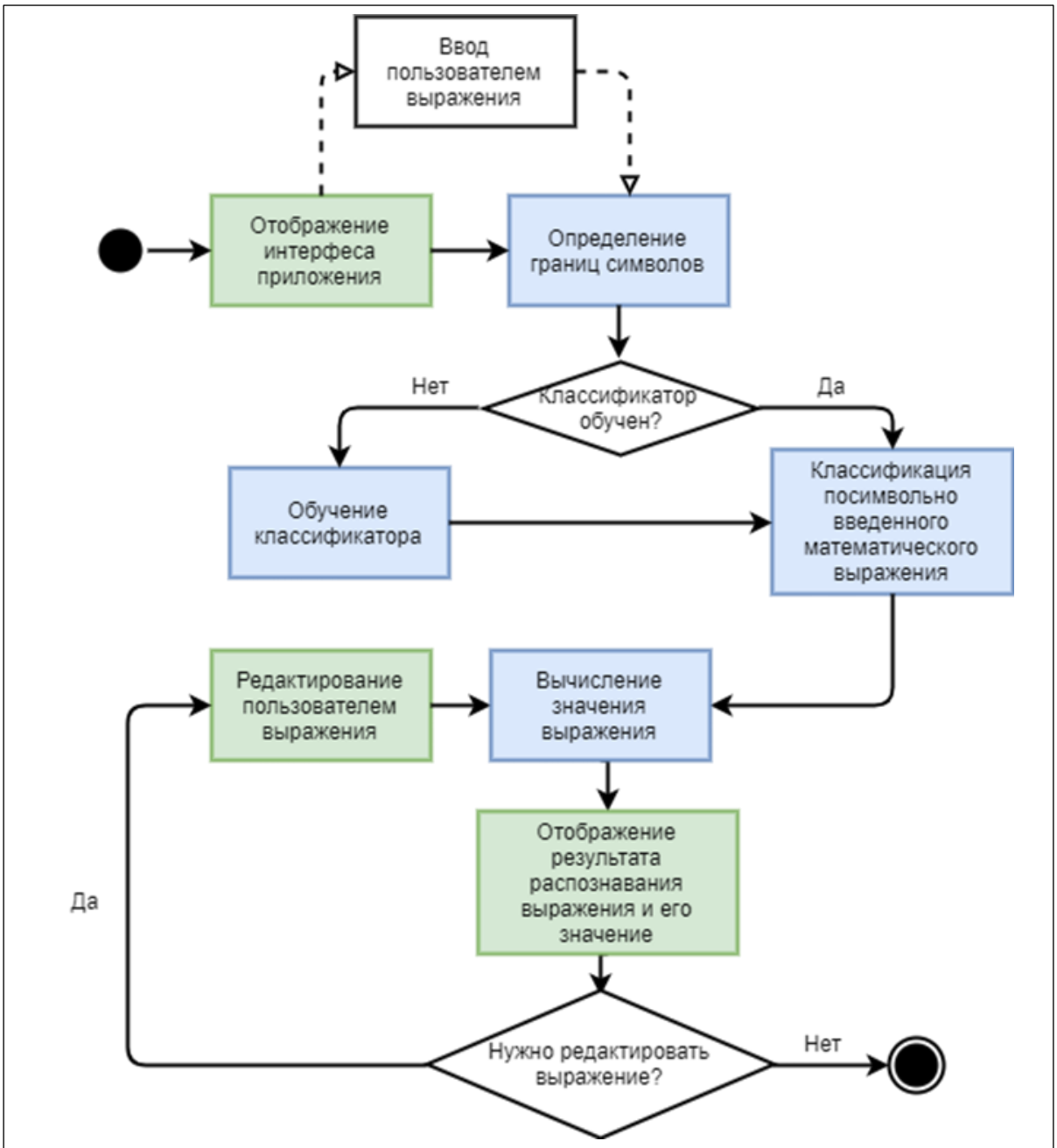


*Рис. 2. Работа приложения с использованием предложенной архитектуры и сценария использования*

***Полученные результаты***

Классификатор нужно обучить таким образом, чтобы он осуществлял классификацию с максимальной точностью. Исследуем каждый из рассмотренных методов классификации, используя в качестве обучающей выборки БД MNIST.

*Мультиномиальная логистическая регрессия.*

Данный метод зависит от параметра $\lambda$, который используется в функции стоимости.

Результаты исследования зависимости точности классификации от выбора параметра $\lambda$ представлены на рисунке 3.

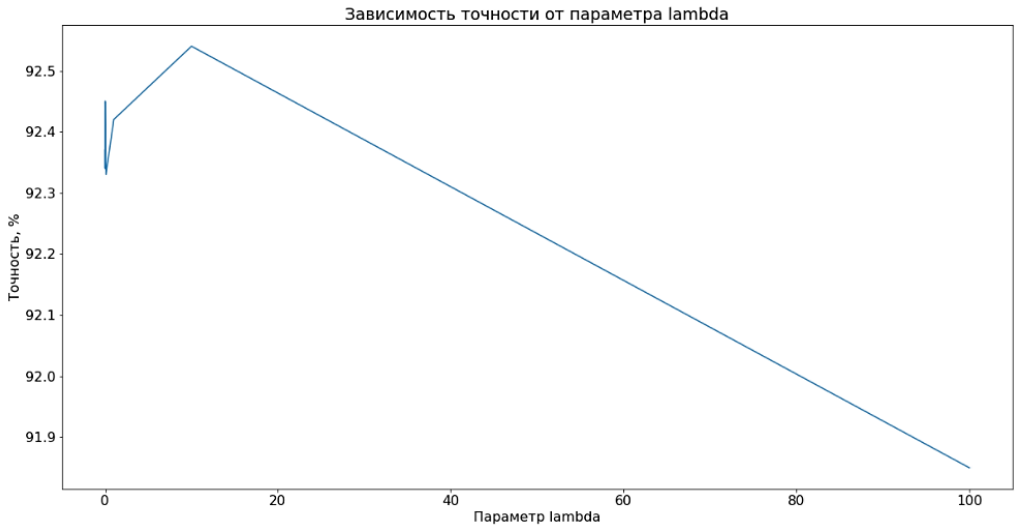


*Рис. 3. Зависимость точности классификации методом мультиномиальной регрессии от выбора параметра $\lambda$*

Достигнута точность 92,54 % 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 исследования зависимости точности классификации от параметра сглаживания представлены на рисунке 4.

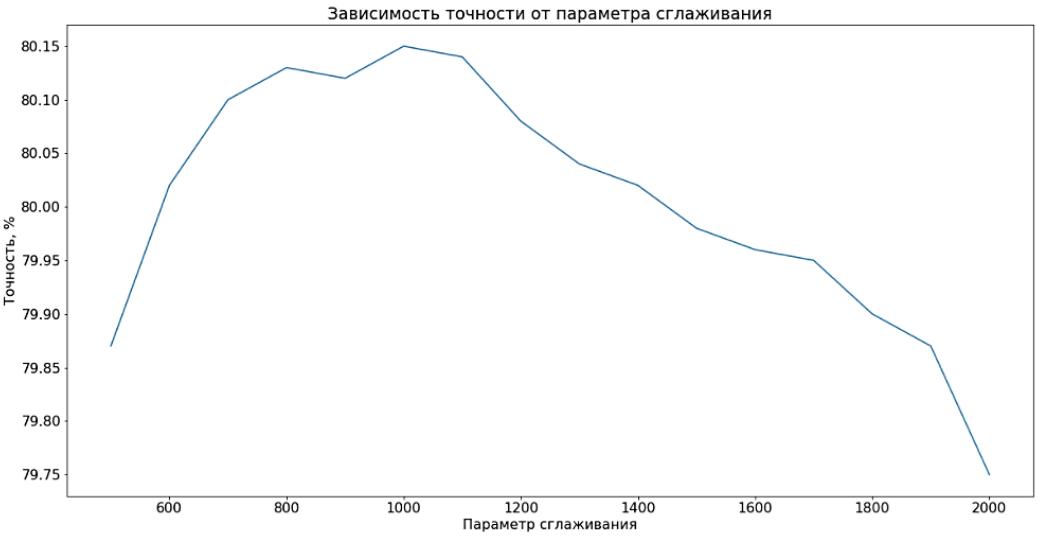


*Рис. 4. Зависимость точности классификации методом наивного Байесовского классификатора от параметра сглаживания*

Достигнута точность 80,15 % при значении параметра сглаживания, равном 1 000.

*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.

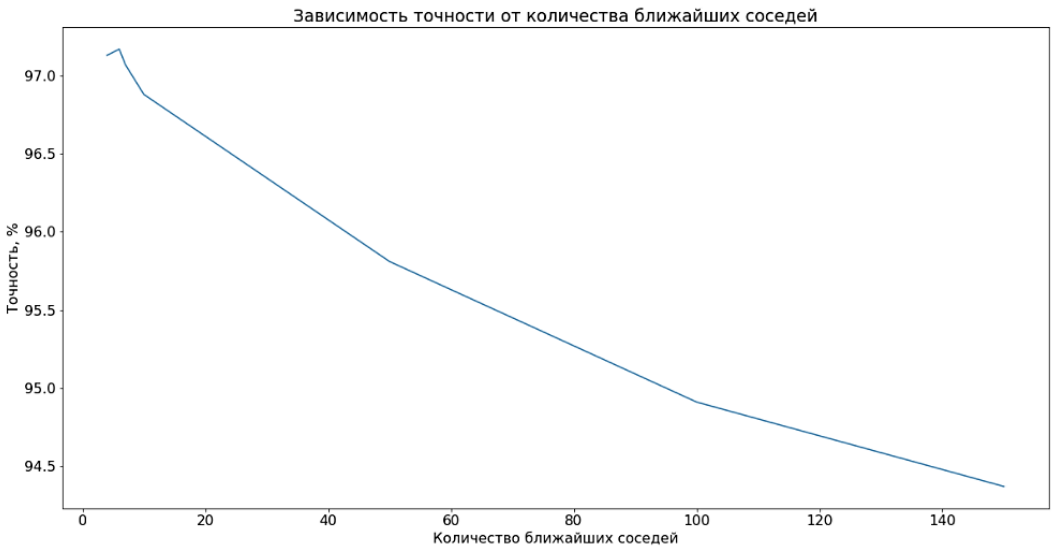


*Рис. 5. Зависимость точности классификации методом k ближайших соседей от количества ближайших соседей*

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

*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.

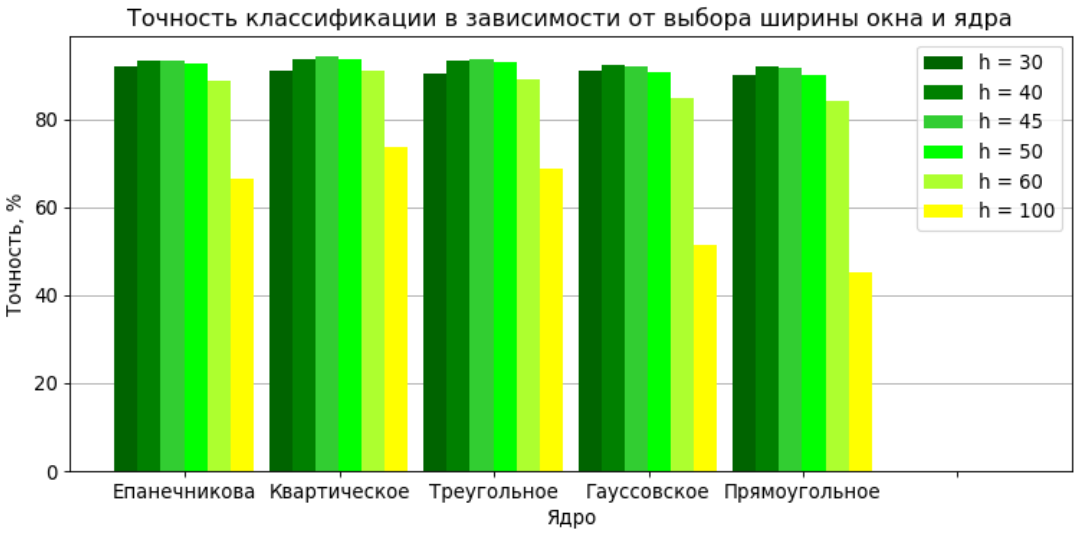


*Рис. 6. Зависимость точности классификации методом Парзеновского окна от ширины окна и вида ядра*

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

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

Данный классификатор зависит от двух параметров – $C$ и $\gamma$.

Результаты исследования зависимости точности классификации от параметров $C$ и $\gamma$ представлены на рисунке 7.

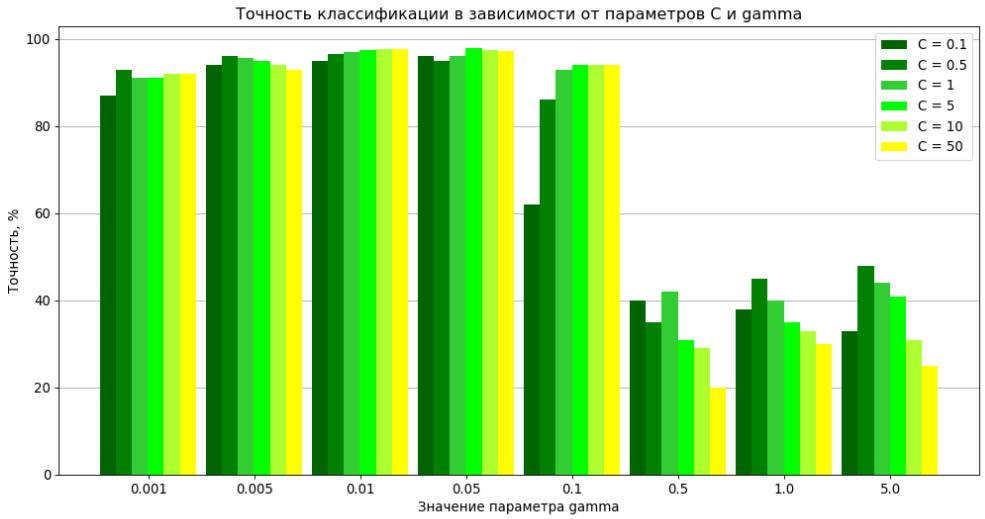


*Рис. 7. График зависимости точности классификации методом опорных векторов от выбранных параметров C и γ*

Достигнута точность 98,52 % при значении параметров $C = 5$ и $\gamma = 0.05$.

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

*Таблица 3*

**Исследование методов классификации**

| **Используемый метод** | **Качество классификации (доля тестовой выборки равна 10 %), %** | **Время классификации 10 000 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, размер которой приблизительно равен 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 примерно 5 секунд.

Метод опорных векторов осуществляет классификацию символа приблизительно за 0,026 секунды 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 БД MNIST с максимальной среди исследованных методов точностью (98,52 %) и высокой скоростью классификации (приблизительно 10 символов за 0,2 секунды).

Решение задачи интерпретации полученного математического выражения решается достаточно просто – использование встроенной функции eval() в языке Python в случае вычислений на сервере или аналогичной функции в языке JavaScript в случае выполнения в браузере.

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

### *Литература*

1. MyScript Calculator. 2015. URL: https://play.google.com/store/apps/details?id=com.visionobjects.calculator&hl=ru (дата обращения: 10.12.2017).

2. Touch Calculator. 2015. URL: https://github.com/zhaorz/Touch-Calculator (дата обращения: 10.12.2017).

3. Mathematical Expression Recognition. 2018. URL: http://cat.prhlt.upv.es/mer/ (дата обращения: 25.12.2017).

4. Hosmer D.W., Lemeshow S. Applied Logistic Regression. Wiley-InterScience Publ., 2000, 375 p.

5. Флах П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных; [пер. с англ. А.А. Слинкина]. М.: ДМК Пресс, 2015. 402 с.

6. Мерков А. Распознавание образов. Введение в методы статистического обучения. М.: Едиториал УРСС, 2011. 256 с.

7. Коэльо Л.П., Ричарт В. Построение систем машинного обучения на языке Python; [пер. с англ. А.А. Слинкина]. М.: ДМК Пресс, 2016. 302 с.

8. The MNIST database. URL: http://yann.lecun.com/exdb/mnist/ (дата обращения: 28.12.2017)

9. Вьюгин В.В. Математические основы теории машинного обучения и прогнозирования. М.: Изд-во МЦНМО, 2013. 390 с.