МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизация производственных процессов»

Методы обработки больших данных в математическом пакете Матлаб

Методические указания
по лабораторным работам
для студентов очной и заочной форм обучения

Ростов-на-Дону
ДГТУ
2023

УДК 681.5

Составитель: Быкадор В.С.

Методические указания. – Ростов-на-Дону : Донской гос. техн. ун-т, 2023. – 22 с.

Методические указания к лабораторным работам по дисциплине «Методы обработки больших данных в математическом пакете Матлаб» предназначены для студентов очной и заочной форм обучения по направлению подготовки 15.04.04 «Автоматизация технологических процессов и производств» профиль «Интеллектуальные системы сбора и анализа больших данных»

УДК 681.5

Печатается по решению редакционно-издательского совета
Донского государственного технического университета

В печать ___.___.20__г.
Формат 60х84/16. Объем ____ усл. п. л.
Тираж___ экз. Заказ № ___.

Издательский центр ДГТУ
Адрес университета и полиграфического предприятия:
344000, г. Ростов-на-Дону, пл. Гагарина, 1

Содержание

Лабораторная работа № 1.
Алгоритм кластеризация методом *K* средних
с применением евклидова расстояния

Цель — реализация алгоритма кластеризации данных методом ***K*** средних, а также исследование обобщающей способности и точности данного алгоритма машинного обучения на различных примерах данных.

Задание

Имеется алгоритм машинного обучения, позволяющий выполнять кластеризацию данных методом ***K*** средних. Данный алгоритм строится на основе метрических моделей машинного обучения. В листинге 1.1 приведен псевдокод алгоритма кластеризации данных методом ***K*** средних.

Листинг 1.1 Алгоритм кластеризации данных методом ***K*** средних

Вход: **данные** $\mathbf{x} \in D$**, число кластеров** K**.**
Выход: **центройды кластеров** $\mu_1,\ \mu_2,\ \ldots,\ \mu_K$ **и кластеры** $D_1,\ D_2,\ \ldots,\ D_K$**.**

/*Случайная инициализация $\mu_1,\ \mu_2,\ \ldots,\ \mu_K$*/
for $j \leftarrow 1$ **to** K **do**
    $\mu_j \leftarrow$ random();
end

/*пересчёт $\mu_1,\ \mu_2,\ \ldots,\ \mu_K$ и разделение $\mathbf{x} \in D$ на кластеры $D_1,\ D_2,\ \ldots,\ D_K$ */
while $\boldsymbol{\mu_1},\ \boldsymbol{\mu_2},\ \ldots,\ \boldsymbol{\mu_K}$ не перестанут изменяться **do**
    for $\mathbf{x}$ **in** D **do**
        $D_j \leftarrow \{\mathbf{x} \in D | \mathbf{x}$ отнесена к кластеру $j \iff \text{argmin}_j \text{Dis}_2(\mathbf{x}, \mu_j)\}$;
    end
    $\mu_j \leftarrow \frac{1}{|D_j|} \sum_{\mathbf{x} \in D_j} \mathbf{x}$;
end

return $\mu_1,\ \mu_2,\ \ldots,\ \mu_K, D_1,\ D_2,\ \ldots,\ D_K$;

1) Необходимо реализовать алгоритм кластеризации данных методом ***K*** средних на языке программирования системы матлаб.

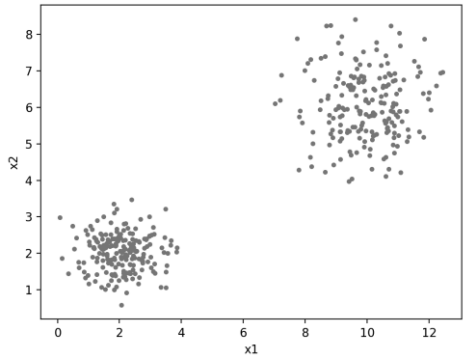
2) Проверить работу машинного кластеризатора на файлах с данными (файлы с данными взять у преподавателя):

lab_metric_data_K_2.dat – два кластера K = 2;

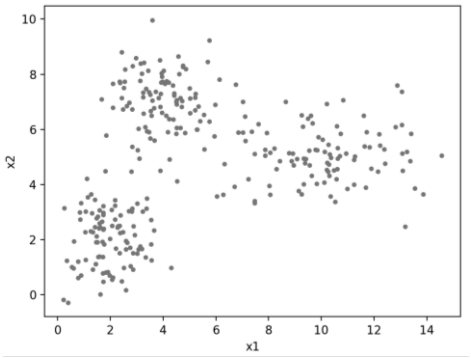
lab_metric_data_K_3.dat – три кластера K = 3, рекомендуется отлаживать алгоритм на данном наборе данных.

lab_metric_data_K_5.dat – пять кластеров K = 5.

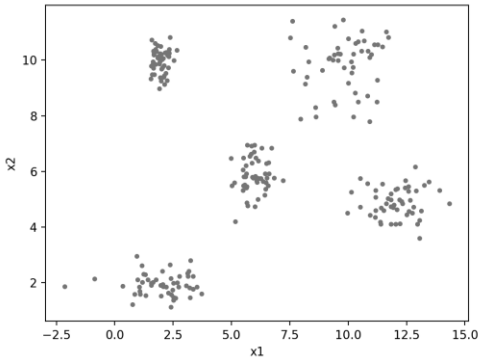
Считанные данные должны выглядеть как показано на рисунке 1.1.



(а)



(б)



(в)

Рис. 1.1 – Исходные данные:
(а) – **lab_metric_data_K_2.dat**; (б) – **lab_metric_data_K_3.dat**; (в) – **lab_metric_data_K_5.dat**

3) По каждому набору данных необходимо показать:

а) первоначальное расположение центройд кластеров $\mu_1,\ \mu_2,\ \ldots,\ \mu_K$, которые заданы случайным образом. Пример показан на рисунке 1.2

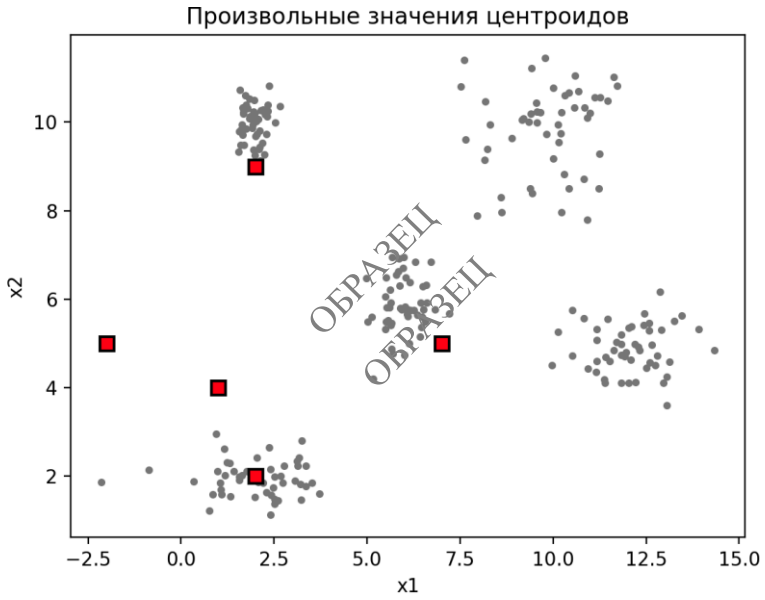


Рис. 1.2 – Первоначальное расположение центройд кластеров, заданное случайно

б) найденные алгоритмом расположения центройд кластеров $\mu_1,\ \mu_2,\ \ldots,\ \mu_K$. Пример показан на рисунке 1.3

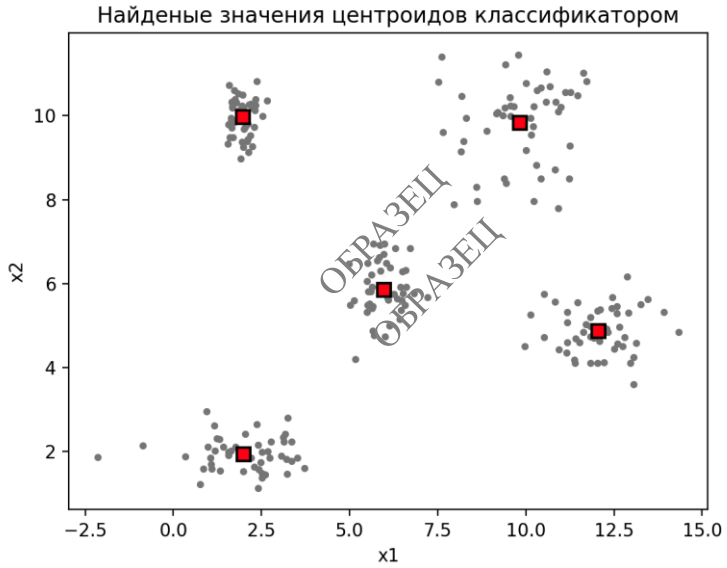


Рис. 1.3 – Найденные алгоритмом метода ***K*** средних расположение центройд кластеров

в) объекты, распределенные алгоритмом метода ***K*** средних по найденным кластерам $D_1, D_2,\ \ldots,\ D_K$. Пример показан на рисунке 1.4

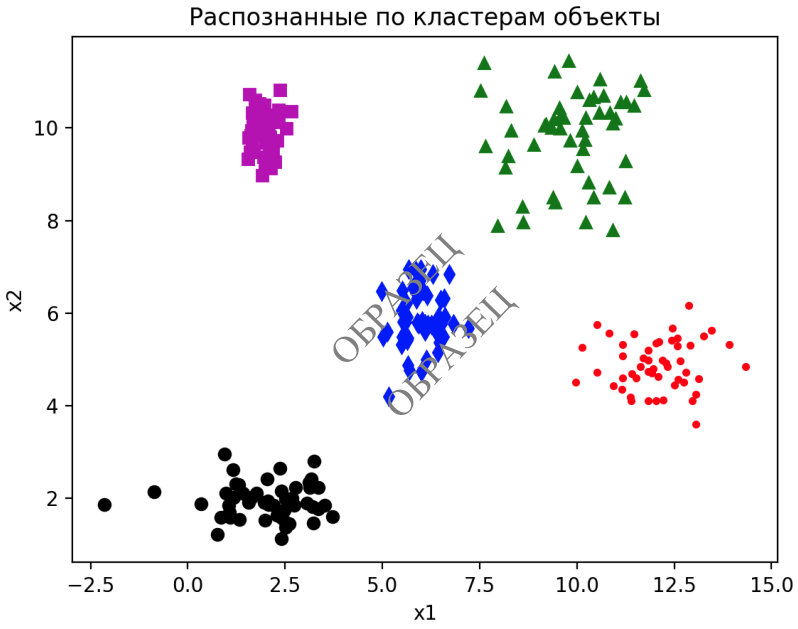


Рис. 1.4 – Объекты, распределенные алгоритмом машинного обучения по кластерам

Лабораторная работа № 2

Алгоритм обучения линейного перцептрона

Цель — реализация и исследование работы алгоритма обучения перцептрона для линейной классификации объектов.

Задание

Имеется алгоритм машинного обучения, позволяющий выполнять линейную классификацию объектов. Данный алгоритм строится на основе обучения перцептрона (простейшей нейронной сети) выполнять линейное разделение двух классов объектов. Перцептрон перебирает обучающий набор данных (это метод машинного обучения с учителем), обновляя вектор весов всякий раз, как встречает неправильно классифицированный пример. В листинге 2.1 приведен псевдокод алгоритма обучения перцептрона как линейного классификатора.

1) Необходимо реализовать алгоритм обучения перцептрона на языке программирования системы матлаб.

2) Для обучения машинного классификатор используются данные из файлов (файлы с данными взять у преподавателя): **lab_perceptron_data_01.dat** … **lab_perceptron_data_05.dat**.

Замечания. Файлы **lab_perceptron_data_01.dat**, **lab_perceptron_data_02.dat**, **lab_perceptron_data_05.dat** содержат хорошо разделяемые классы данных, перцептрон можно будет обучить на одном и том же значении η.

Листинг 2.1 Алгоритм обучения линейного перцептрона

Вход: **помеченные входные данные $\mathbf{x} \in \boldsymbol{D}$,**
скорость обучения $\boldsymbol{\eta}$,
величина смещения ϑ [*)]**.**
Выход: **вектор весов $\mathbf{w}$ для классификатора $\hat{y} = \text{sign}(\mathbf{w} \cdot \mathbf{x})$.**

```
t ← 0;        /*счетчик эпох*/

/*Инициализация вектора весов нулями, в общем можно инициализировать
и другими значениями*/
w ← 0;

/*флаг сходимости алгоритма и завершения главного цикла*/
converged ← false;

while converged = false do
    /*если количество эпох больше какого-то числа, то алгоритм не
    сошёлся – решение не найдено*/
    t ← t + 1;
    if t ≥ 100
    then
        error;
    end

    /*коррекция вектора весов w перцептрона*/
    converged ← true;
    for i ← 1 in |D| do
        if y_i · w · x_i ≤ 0        /*т.е. ŷ ≠ y*/
        then
            w ← w + η · y_i · x_i;
            ϑ ← ϑ + η · y_i;  *)
            /*меняли w, значит алгоритм еще не сошёлся*/
            converged ← false;
        end
    end
end
return w, ϑ *);
```

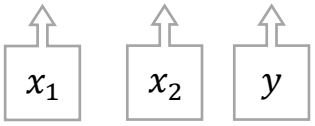
*) *Данные части алгоритма необходимы только для последующего графического приближенного отображения решающей границы, для визуального контроля классификации двумерных данных. Классический алгоритм обучения перцептрона используется только для отыскания вектора весов* **w** *классификатора.*

Файл **lab_perceptron_data_03.dat**, содержит плохо разделимые классы данных, поэтому, возможно, алгоритм не сойдётся.

Файл **lab_perceptron_data_04.dat**, содержит близко расположенные друг к другу классы данных, поэтому скорей всего для их разделения потребуется отдельный подбор значения скорости обучения из диапазона $0 < \eta \leq 1$ и для более правильного отображения решающей границы возможно придется подобрать величину смещения ϑ.

Формат данных. Данные, содержащиеся в файлах **lab_perceptron_data_XX.dat** представляют собой массив следующего формата:

```
array([
    array([1.23, 0.2, 1]),
    array([2.0, 1.5, 1]),
.....................................................................
    array([0.8, 1.3, -1]),
    array([3.5, 2.3, -1])
])
```



x_1, x_2 – координаты точек по оси абсцисс и ординат соответственно, то есть значения двух признаков объекта. Так как рассматривается метод машинного обучения с учителем, а данные в файлах являются обучающими наборами, то каждый объект содержит метку класса y, которая принимает одно из двух значений $y\ \epsilon\ \{+1, -1\}$. То есть объект относится к классу +1 или классу -1.

Объекты класса +1 можно показать на графике как “ ° ”, а объекты класса -1 можно показать на графике как “ × “, как показано на рисунке 2.1.

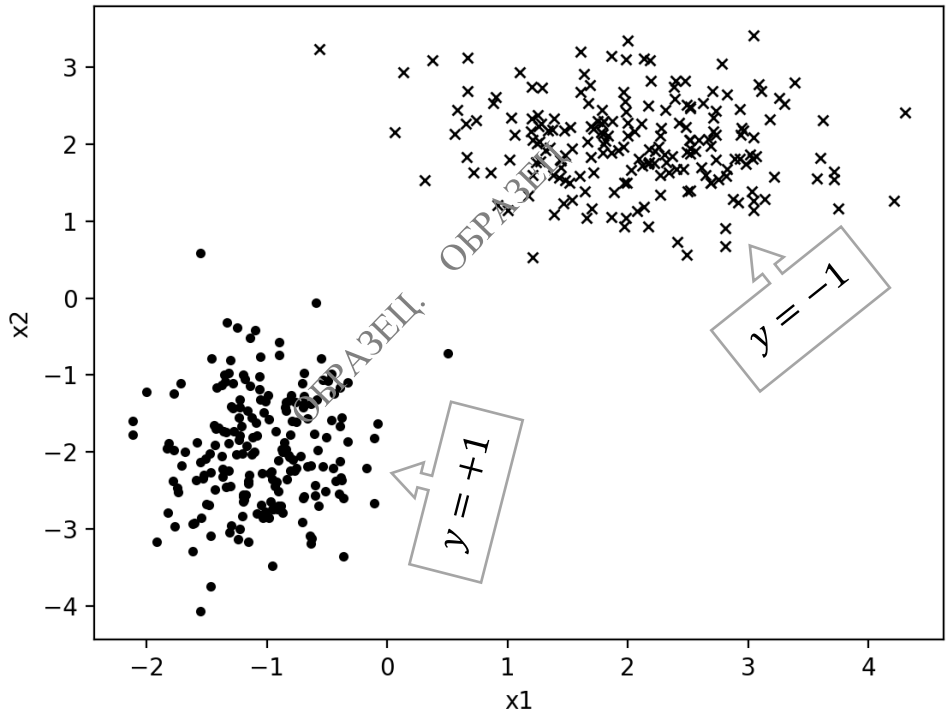


Рис. 2.1 – Пример считанных данных из файла **lab_perceptron_data_01.npy**

3) Проверить работу классификатора, для этого необходимо:

а) на каждом графике, кроме того на котором алгоритм не сошёлся, показать графически решающую границу. Если алгоритм машинного обучения рассчитал вектор весов $\mathbf{w} = \{\boldsymbol{w_0} \quad \boldsymbol{w_1}\}^{\boldsymbol{T}}$ и скорректировал смещение ϑ, то приблизительную решающую границу можно найти по выражению (2.1):

$$line(x_1) \approx \frac{-(w_0 \cdot x_1 + \vartheta)}{w_1} \tag{2.1}$$

На рисунке 2.2 показана приблизительная решающая граница для данных из файла **lab_perceptron_data_01.npy**.

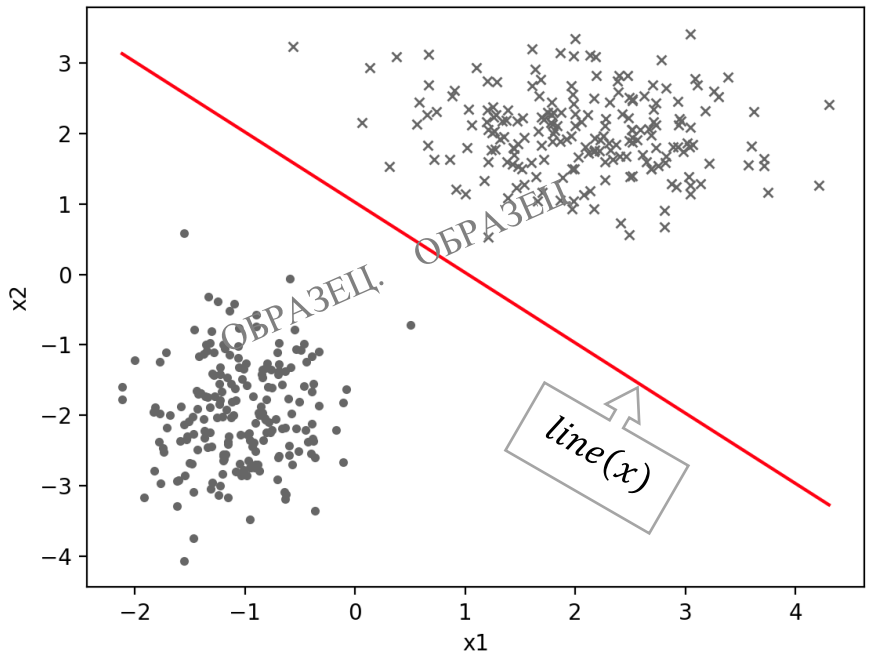


Рис. 2.2 – Приблизительная решающая граница для данных из файла **lab_perceptron_data_01.dat**

б) задаться разными точками «сверху» и «снизу» приближенной решающей границы и по выражению линейного классификатора (2.2) найти класс, к которому относиться точка

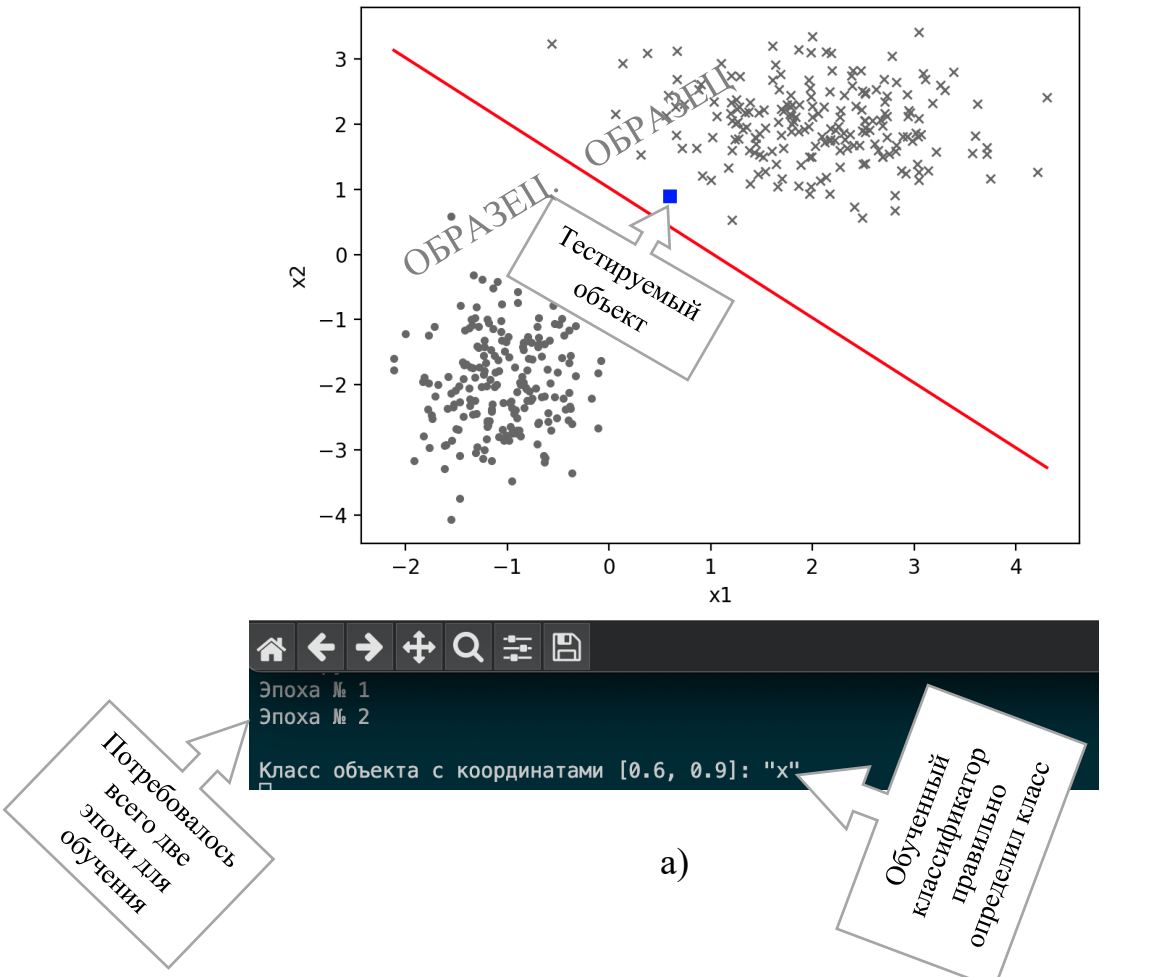
$$\hat{y} = \text{sign}(\mathbf{w} \cdot \mathbf{x}) \qquad (2.2)$$

где $\mathbf{w} = \{w_0 \quad w_1\}^T$ – вектор весов, найденный перцептроном;

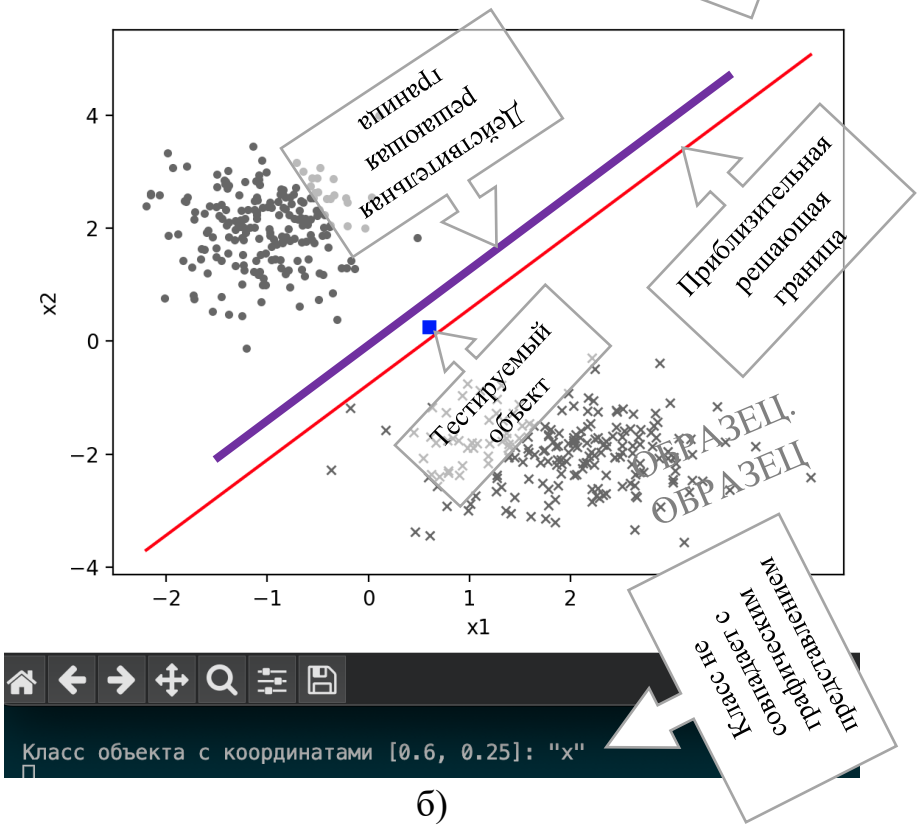
$\mathbf{x} = \{x_0 \quad x_1\}^T$ – координаты тестируемого объекта (точки).

На рисунке 2.3 показаны примеры тестирования. Для удобства классификатор возвращает значения классов не как $y \in \{+1, -1\}$, а как $y \in \{\circ, \times\}$.

На рисунке 2.3, а) показано совпадение найденного классификатором класса и графическим расположением объекта на плоскости. На рисунке 2.3, б) показана фиктивная ошибка классификатора, в действительности классификатор определил класс верно (объект ближе находится к классу «×»). Ошибка заключается в приближенном отображении решающей границе, которая сдвинута в сторону объектов класса «×».



а)



б)

Рис. 2.3 – Примеры тестирования:
а) без ошибки классификации; б) с фиктивной ошибкой классификации

Лабораторная работа № 3

Алгоритм обучения ядерного перцептрона

Цель — реализация и исследование работы алгоритма обучения ядерного перцептрона с полиномиальным ядром.

Задание

Имеется алгоритм машинного обучения, позволяющий выполнять бинарную классификацию объектов с нелинейной решающей границей. Данный алгоритм строится на основе обучения ядерного перцептрона выполнять разделение двух классов объектов, имеющих нелинейную решающую границу. Перцептрон перебирает обучающий набор данных (это метод машинного обучения с учителем), обновляя вектор множителей Лагранжа. В листинге 3.1 приведен псевдокод алгоритма обучения ядерного перцептрона как бинарного классификатора.

1) Необходимо реализовать алгоритм обучения ядерного перцептрона (см. листинг 3.1) на языке программирования системы матлаб.

2) Для обучения машинного классификатора используются данные из файла (файл с данными взять у преподавателя): **lab_kernel_perceptron_data.csv**.

Объекты класса +1 можно показать на графике как “ ■ ”, а объекты класса -1 можно показать на графике как “ ° “, как показано на рисунке 3.1.

В качестве ядра перцептрона использовать полиномиальное ядро:

$$\boldsymbol{\kappa}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \left(\mathbf{x}_i \cdot \mathbf{x}_j + 1\right)^p$$

Листинг 3.1 Алгоритм обучения ядерного перцептрона

Вход: **помеченные входные данные $x \in D$, ядерная функция $\kappa(\mathbf{x}_i, \mathbf{x}_j)$.**
Выход: **множители α_i, определяющих нелинейную решающую границу для классификатора $\hat{y}(\mathbf{x}_i) = \text{sign}(\mathbf{w}(\mathbf{x}_i))$.**

$\alpha_i \leftarrow 0$ *для* $1 \leq i \leq |D|$; */*инициализация вектора множителей нулями*/*
$converged \leftarrow$ ***false***; */*флаг сходимости алгоритма и завершения цикла*/*

$t \leftarrow 0$; /*счетчик эпох*/

while $converged = false$ **do**
/*коррекция вектора множителей α_i*/
$converged \leftarrow true$;

/*если количество эпох больше какого-то числа, то алгоритм не сошёлся – прервать цикл*/
$t \leftarrow t + 1$;
if $t \geq 30$
then
break;
end

for $i \leftarrow 1$ **in** $|D|$ **do**
if $y_i \cdot \left(\sum_{j=1}^{|D|} \alpha_j y_j \cdot \kappa(\mathbf{x}_i, \mathbf{x}_j)\right) \leq 0$
then
$\alpha_i \leftarrow \alpha_i + 1$;
/*меняли α_i, значит алгоритм еще не сошёлся*/
$converged \leftarrow false$;
end
end
end
return α_i;

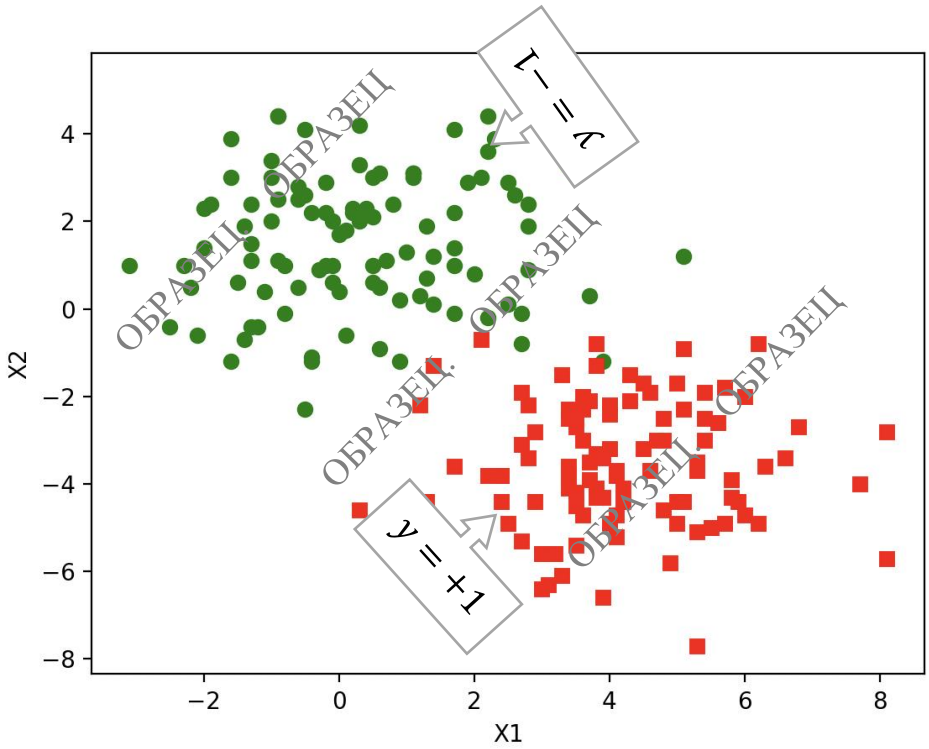


Рис. 3.1 – Отображение данных их файла **lab_kernel_perceptron_data.csv**

Формат данных. Данные, содержащиеся в файле **lab_kernel_perceptron_data.csv** представляют собой столбцы с признаками объекта $\mathbf{X} = \{x_1 \quad x_2\}$ и меткой класса $y \in \{-1 \quad +1\}$ (см. рис. 3.2) (разделитель данных – символ `'\t'`).

lab_kernel_perceptron_data.csv

	x1	x2	y
1			
2			
3	-0.5	-2.3	-1
4	-2.2	0.5	-1
5	8.1	-2.8	1
6	1.1	3.1	-1
7	-0.1	2.0	-1
8	3.5	-4.5	1

Рис. 3.2 – Формат файла данных

3) Отладку и проверку алгоритма обучения персептрона можно выполнять со степенью полиномиального ядра $p = 2$:

$$\boldsymbol{\kappa}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^2$$

При этом, параметры классифицируемого объекта (координаты) можно взять в глубине обучающих классов:

$\mathbf{X} = \{0{,}95 \quad 0{,}80\}$ – класс «-1» (класс «кружки»)

$\mathbf{X} = \{4{,}30 \quad -2{,}64\}$ – класс «+1» (класс «квадратики»)

Добиваясь от перцептрона правильной классификации тестируемого объекта (см. рисунок 3.3).

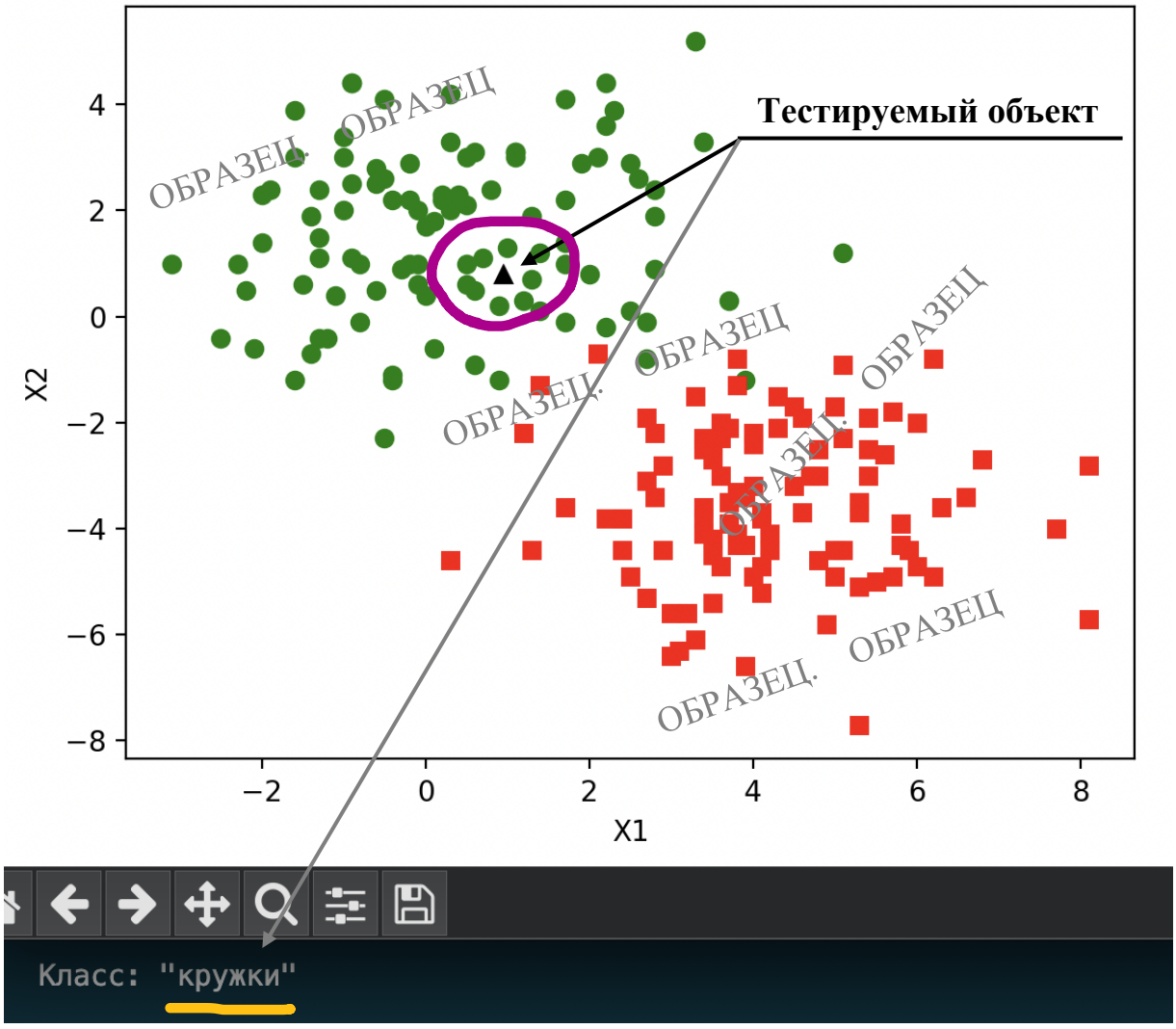


Рис. 3.3 – Пример классификации тестируемого объекта ядерным перцептроном

Лабораторная работа № 4

Алгоритм обучения решающего дерева

Цель — реализация и исследование работы алгоритма обучения решающего дерева.

Задание

Требуется реализовать алгоритм машинного обучения решающего дерева, которое должно выполнять бинарную классификацию объектов. Данный алгоритм строится на основе последовательного разделения множества размеченных объектов на подмножества для формирования логических выражений отнесения тестового объекта к соответствующему классу по совокупности входных данных. Эти логические (конъюнктивные) выражения представлены в виде дерева – решающего дерева.

Выполнить машинное обучение решающего дерева означает получить древовидную структуру признаков и фактов из размеченного обучающего набора, который представлен в виде таблицы. Для машинного обучения решающего дерева требуется реализовать соответствующий алгоритм.

1) Необходимо реализовать алгоритм обучения решающего дерева (см. листинг 4.1) на языке программирования системы матлаб.

Листинг 4.1 Алгоритм обучения решающего дерева

Сигнатура функции: $\boldsymbol{GrowTree(D, F)}$

Вход: **помеченные входные данные** D**, множество признаков** F.

Выход: **дерево признаков (узлов)** T **с помеченными листьями** L.

$T \leftarrow linkedlist(\ );$ */*связанный список структур*/*

$L \leftarrow \boldsymbol{Label(D)};$ */*возвращаем класс объектов для множества*/*

If $\boldsymbol{L} = None$ **then** /*класс для множества не определён*/

$S \leftarrow \boldsymbol{BestSplit(D, F)}$; /*возвращает наиболее чистый признак для разделения множества *D**/

If $S = None$ **then return**; **/*все признаки уже рассмотрены => ошибка и выход */**

$T \leftarrow linkedlist(S);$ /*добавили узел в дерево к родительскому узлу*/

разделяем множество D *на подмножества* D_i *в соответствии с литералами в* S; */*требуется реализовать*/*

for $\boldsymbol{D_i}$ **in** $\boldsymbol{D}$ **do**

$\boldsymbol{GrowTree(D_i, F)};$ **/*рекурсивный вызов функции*/**

end

else

$T \leftarrow linkedlist(L);$ /*добавили лист к родительскому узлу*/

end

return $\boldsymbol{T}$**;** /*дерево построено*/

Сигнатура функции: $\boldsymbol{BestSplit(D, F)}$

Вход: **помеченные входные данные** D**, множество признаков** F.

Выход: признак f**, по которому выполнять разделение.**

$I_{min} \leftarrow 1;$

$f_{best} \leftarrow None;$

for f **in** F **do**

/*требуется реализовать*/

разделить D *на подмножества* $D_1, \ldots, D_m$ *согласно значениям* v_j *признака* f*;*

/*т. е. необходимо для каждого значения признака найти разделения

$f = [v_1, v_2, v_3]$ => [n_1+, n_1-] [n_2+, n_2-] [n_3+, n_3-]

(Длина = {1, 2, 3} => [3+, 1-] [1+, 0] [1+, 4-])

*/

If $\mathrm{Imp}(\{\boldsymbol{D_1}, \boldsymbol{D_2}, \ldots, \boldsymbol{D_m}\}) < \boldsymbol{I_{min}}$ **then**

$I_{min} \leftarrow \mathrm{Imp}(\{D_1, D_2, \ldots, D_m\});$

$f_{best} \leftarrow f;$

end

/*

т.к. необходимо сохранять признаки по которым ранее было выполнено разделение, то могут быть значения типа n(+) = 0, n(-) = 0 => n(+) + n(-) = 0 => деление на нуль

в таких случаях требуется пропуск итерации

*/

end

return $\boldsymbol{f_{best}}$**;**

Сигнатура функции: $Label(D)$
Вход: **помеченные входные данные** D.
Выход: **возвращает один из классов или** $None$.

```
for D_i in D do
    /*проверяем класс каждого D_i объекта с классов первого объекта D_1*/
    If ĉ(D_i) ! = ĉ(D_1) then
        return None; /*есть хотя бы одни D_i с классом отличным от других объектов из D*/
    end
end
return ĉ(D_1); /*класс для объектов всего множества определён*/
```

2) Для обучения решающего дерева используются данные из файла (файл с данными взять у преподавателя): **lab_decision_tree.csv**.

Легенда данных, следующая: у страхового агента, занимающегося страховкой транспортных средств есть некоторая статистическая информация, которую он накопил в результате своей деятельности. Эта информация представляет собой набор данных по страховке транспортных средств клиентов или отказах в страховке. В таблице 4.1 представлены эти статистические данные.

Таблица № 4.1. Статистические данные страхового агента

Возраст водителя, лет	Место эксплуатации ТС	Сведения о ДТП за последний год	Стаж вождения, лет	Тип авто	Решение о выдаче страховки
больше 40	город	есть	больше 10	обычное	страховать
больше 40	город	есть	меньше 10	обычное	отказать
больше 40	город	нет	меньше 10	обычное	страховать
меньше 40	город	нет	меньше 10	обычное	страховать
меньше 40	с/х местность	нет	меньше 10	обычное	отказать
меньше 40	город	нет	больше 10	спортивное	страховать
больше 40	с/х местность	есть	больше 10	обычное	страховать
меньше 40	с/х местность	есть	меньше 10	спортивное	отказать
больше 40	город	нет	больше 10	спортивное	страховать
меньше 40	с/х местность	есть	меньше 10	обычное	отказать

Для обучения решающего дерева, с последующей программной реализацией алгоритма машинного обучения, данные из таблицы № 4.1 были закодированы следующим образом:

Названия признаков:

Возраст водителя	⇨	age
Место эксплуатации ТС	⇨	location
Сведения о ДТП за последний год	⇨	accident
Стаж вождения	⇨	experience

Тип авто ⇨ typecar
Решение о выдаче страховки ⇨ y

Коды значений признаков:

Возраст водителя, лет
больше 40 ⇨ 1
больше 40 ⇨ 0

Место эксплуатации ТС
город ⇨ 1
с/х местность ⇨ 0

Сведения о ДТП за последний год
есть ⇨ 1
нет ⇨ 0

Стаж вождения, лет
больше 10 ⇨ 1
меньше 10 ⇨ 0

Тип авто
спортивное ⇨ 1
обычное ⇨ 0

Решение о выдаче страховки
страховать ⇨ 1
отказать ⇨ 0

После замены признаков и из значений кодами, таблица № 4.1 преобразуется к виду, показанному в таблицы № 4.2.

Таблица № 4.2. Статистические данные страхового агента в закодированном виде

age	location	accident	experience	typecar	y
1	1	1	1	0	1
1	1	1	0	0	0
1	1	0	0	0	1
0	1	0	0	0	1
0	0	0	0	0	0
0	1	0	1	1	1
1	0	1	1	0	1
0	0	1	0	1	0
1	1	0	1	1	1
0	0	1	0	0	0

Данные из таблицы № 4.2 приведены в файле **lab_decision_tree.csv**

Перечень использованных информационных ресурсов

1. Плохотников, К. Э. Методы разработки математических моделей и вычислительный эксперимент на базе пакета Matlab: Курс лекций, Москва: Издательство "СОЛОН-Пресс", 2017
2. Алибеков, И.Ю. Теория вероятностей и математическая статистика в среде MATLAB: учебное пособие Санкт-Петербург: Лань, 2019
3. Тимохин, А. Н., Р. Ю. Моделирование систем управления с применением Matlab: Учебное пособие Москва: ООО "Научно-издательский центр ИНФРА-М", 2020
4. Макшанов, А.В., Журавлев, А.Е. Большие данные. Big Data Санкт-Петербург: Лань, 2021
5. Романов, П.С., Романова, И.П. Системы искусственного интеллекта. Моделирование нейронных сетей в системе MATLAB. Лабораторный практикум Санкт-Петербург: Лань, 2021