**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра САПР**

**ОТЧЕТ**

**по лабораторной работе №1**

**по дисциплине «Автоматизация схемотехнического проектирования»**

**Тема: генерация модельных наборов данных**

|  |  |  |
| --- | --- | --- |
| Студент гр. 1302 |  | Новиков Г.В. |
| Студентка гр. 1302 |  | Романова О.В. |
| Студентка гр. 1302 |  | Марзаева В.И. |
| Преподаватель |  | Боброва Ю.О. |

Санкт-Петербург

2025

**Цель работы**

Получение навыков работы с numpy-массивами и написание функций на языке Python на примере генерации массивов произвольно распределенных данных.

Для изучения различных классификаторов, их свойств и особенностей, создайте модельные данные, форму распределения и смешанность которых можно регулировать вручную.

Наборы формируются так, чтобы первый набор удовлетворял условиям применения текущего метода классификации/кластеризации, второй – нет. Вы можете поэкспериментировать и создать больше тестовых наборов для исследования ограничений метода. Также вы увидите, какие оценки эффективности наилучшим образом отвечают на вопрос – подходит ли выбранный метод для анализа имеющихся данных.

Создайте новую функцию внутри файла DataGenerator.py, назовите ее nonlinear\_dataset\_N, где N – номер вашего варианта. Данная функция должна генерировать двумерный массив данных, распределенный в пространстве заданным образом. Форма приведена в приложении А. Для ее создания используйте любые математические методы генерации данных. Обеспечьте максимально близкое воспроизведение заданной формы.

**Ход работы**

1. *Полный код программы:*

*lab1.py:*

import numpy as np

import matplotlib.pyplot as plt

from data\_generator import norm\_dataset, nonlinear\_dataset\_13

mu0 = [0, 1, 1]

mu1 = [5, 5, 5]

sigma0 = [1, 1, 2]

sigma1 = [1, 2, 1]

N = 1000 # число объектов класса

col = len(mu0) # количество столбцов-признаков – длина массива средних

mu = [mu0, mu1]

sigma = [sigma0, sigma1]

X, Y, class0, class1 = norm\_dataset(mu, sigma, N)

# X, Y, class0, class1 = nonlinear\_dataset\_13([0, 0, 0], [0, 0, 0], [6, 0, 2], [2, 0, 0], N)

# разделяем данные на 2 подвыборки

trainCount = round(0.7\*N\*2) # не забываем округлить до целого

Xtrain = X[0:trainCount]

Xtest = X[trainCount:N\*2+1]

Ytrain = Y[0:trainCount]

Ytest = Y[trainCount:N\*2+1]

# построение гистограмм распределения для всех признаков

for i in range(0, col):

\_ = plt.hist(class0[:, i], bins='auto', alpha=0.7) # параметр alpha позволяет задать прозрачность цвета

\_ = plt.hist(class1[:, i], bins='auto', alpha=0.7)

plt.title('Parameter ' + str(i))

plt.xlabel('Parameter value')

plt.ylabel('Number of objects')

plt.savefig('hist\_' + str(i + 1) + '.png') # сохранение изображения в файл

plt.show()

# построение одной скатеррограммы по выбранным признакам

plt.scatter(class0[:, 0], class0[:, 2], marker=".", alpha=0.7)

plt.scatter(class1[:, 0], class1[:, 2], marker=".", alpha=0.7)

plt.title('Scatter')

plt.xlabel('Parameter 0')

plt.ylabel('Parameter 2')

plt.savefig('scatter\_' + str(i + 1) + '.png')

plt.show()

*data\_generator.py:*

import numpy as np

def norm\_dataset(mu,sigma,N):

mu0 = mu[0]

mu1 = mu[1]

sigma0 = sigma[0]

sigma1 = sigma[1]

col = len(mu0) # количество столбцов-признаков – длина массива средних

class0 = np.random.normal(mu0[0], sigma0[0], [N, 1]) # инициализируем первый столбец (в Python нумерация от 0)

class1 = np.random.normal(mu1[0], sigma1[0], [N, 1])

for i in range(1, col):

v0 = np.random.normal(mu0[i], sigma0[i], [N, 1])

class0 = np.hstack((class0, v0))

v1 = np.random.normal(mu1[i], sigma1[i], [N, 1])

class1 = np.hstack((class1, v1))

Y1 = np.ones((N, 1), dtype=bool)

Y0 = np.zeros((N, 1), dtype=bool)

X = np.vstack((class0, class1))

Y = np.vstack((Y0, Y1)).ravel() # ravel позволяет сделать массив плоским – одномерным, размера (N,)

# перемешиваем данные

rng = np.random.default\_rng()

arr = np.arange(2\*N) # индексы для перемешивания

rng.shuffle(arr)

X = X[arr]

Y = Y[arr]

return X, Y, class0, class1

def nonlinear\_dataset\_13(cen0, cen1, radii0, radii1, N):

col = len(cen0)

theta = 2 \* np.pi \* np.random.rand(N)

theta = theta[:, np.newaxis]

class0 = np.empty((N, col))

class1 = np.empty((N, col))

r = radii0[0] + np.random.rand(N)

r = r[:, np.newaxis]

class0[:, 0] = (r \* np.sin(theta) + cen0[0]).flatten()

r = radii1[0] + np.random.rand(N)

r = r[:, np.newaxis]

class1[:, 0] = (r \* np.sin(theta) + cen1[0]).flatten()

for i in range(1, col):

r = radii0[i] + np.random.rand(N)

r = r[:, np.newaxis]

class0[:, i] = (r \* np.cos(theta) + cen0[i]).flatten()

r = radii1[i] + np.random.rand(N)

r = r[:, np.newaxis]

class1[:, i] = (r \* np.cos(theta) + cen1[i]).flatten()

Y1 = np.ones((N, 1), dtype=bool)

Y0 = np.zeros((N, 1), dtype=bool)

X = np.vstack((class0, class1))

Y = np.vstack((Y0, Y1)).ravel() # ravel позволяет сделать массив плоским – одномерным, размера (N,)

# перемешиваем данные

rng = np.random.default\_rng()

arr = np.arange(2\*N) # индексы для перемешивания

rng.shuffle(arr)

X = X[arr]

Y = Y[arr]

return X, Y, class0, class1

1. *Пояснения к коду:*

lab1.py — основной скрипт, который генерирует данные, разделяет их на обучающую и тестовую выборки, строит гистограммы и scatter plot.

data\_generator.py — файл с функциями для генерации линейных и нелинейных данных.

*lab1.py:*

*Импорт библиотек:*

import numpy as np

import matplotlib.pyplot as plt

from data\_generator import norm\_dataset, nonlinear\_dataset\_13

numpy — библиотека для работы с массивами и математическими операциями.

matplotlib.pyplot — библиотека для визуализации данных (графики, гистограммы и т.д.).

norm\_dataset и nonlinear\_dataset\_13 — функции из файла data\_generator, которые генерируют наборы данных.

*Задание параметров для генерации данных:*

mu0 = [0, 1, 1]

mu1 = [5, 5, 5]

sigma0 = [1, 1, 2]

sigma1 = [1, 2, 1]

mu0 и mu1 — средние значения для двух классов.

sigma0 и sigma1 — стандартные отклонения для двух классов.

*Настройка параметров генерации:*

N = 1000

col = len(mu0)

mu = [mu0, mu1]

sigma = [sigma0, sigma1]

N — количество объектов в каждом классе.

col — количество признаков (столбцов) в данных.

mu и sigma — списки, содержащие параметры для двух классов.

*Генерация данных:*

X, Y, class0, class1 = norm\_dataset(mu, sigma, N)

X — матрица признаков (объекты и их признаки).

Y — метки классов (0 или 1).

class0 и class1 — данные для каждого класса отдельно.

*Разделение данных на обучающую и тестовую выборки:*

trainCount = round(0.7\*N\*2)

Xtrain = X[0:trainCount]

Xtest = X[trainCount:N\*2+1]

Ytrain = Y[0:trainCount]

Ytest = Y[trainCount:N\*2+1]

Данные делятся в соотношении 70% на обучение и 30% на тестирование.

*Построение гистограмм распределения признаков:*

for i in range(0, col):

\_ = plt.hist(class0[:, i], bins='auto', alpha=0.7)

\_ = plt.hist(class1[:, i], bins='auto', alpha=0.7)

plt.title('Parameter ' + str(i))

plt.xlabel('Parameter value')

plt.ylabel('Number of objects')

plt.savefig('hist\_' + str(i + 1) + '.png')

plt.show()

Для каждого признака строятся гистограммы распределения для двух классов.

Гистограммы сохраняются в файлы ‘hist\_1.png’, ‘hist\_2.png’ и т.д.

*Построение скатеррограммы (scatter plot):*

plt.scatter(class0[:, 0], class0[:, 2], marker=".", alpha=0.7)

plt.scatter(class1[:, 0], class1[:, 2], marker=".", alpha=0.7)

plt.title('Scatter')

plt.xlabel('Parameter 0')

plt.ylabel('Parameter 2')

plt.savefig('scatter\_' + str(i + 1) + '.png')

plt.show()

Строится график рассеяния для двух выбранных признаков (0 и 2).

График сохраняется в файл ‘scatter\_1.png’.

*data\_generator.py:*

*Функция norm\_dataset:*

def norm\_dataset(mu, sigma, N):

Генерирует данные с нормальным распределением.

*Инициализация параметров:*

mu0 = mu[0]

mu1 = mu[1]

sigma0 = sigma[0]

sigma1 = sigma[1]

col = len(mu0)

Извлекаются параметры для двух классов.

*Генерация данных для каждого класса:*

class0 = np.random.normal(mu0[0], sigma0[0], [N, 1])

class1 = np.random.normal(mu1[0], sigma1[0], [N, 1])

for i in range(1, col):

v0 = np.random.normal(mu0[i], sigma0[i], [N, 1])

class0 = np.hstack((class0, v0))

v1 = np.random.normal(mu1[i], sigma1[i], [N, 1])

class1 = np.hstack((class1, v1))

Для каждого признака генерируются данные с нормальным распределением.

Данные объединяются в матрицы class0 и class1.

*Создание меток классов:*

Y1 = np.ones((N, 1), dtype=bool)

Y0 = np.zeros((N, 1), dtype=bool)

Y1 — метки для класса 1 (все значения True).

Y0 — метки для класса 0 (все значения False).

*Объединение данных и меток:*

X = np.vstack((class0, class1))

Y = np.vstack((Y0, Y1)).ravel()

Данные и метки объединяются в общие массивы X и Y.

*Перемешивание данных:*

rng = np.random.default\_rng()

arr = np.arange(2\*N)

rng.shuffle(arr)

X = X[arr]

Y = Y[arr]

*Функция nonlinear\_dataset\_13:*

def nonlinear\_dataset\_13(cen0, cen1, radii0, radii1, N):

Генерирует нелинейные данные (данные, распределенные по окружности).

cen0, cen1 – координаты центров окружностей.

radii0, radii1 – радиусы окружностей.

*Инициализация параметров:*

col = len(cen0)

theta = 2 \* np.pi \* np.random.rand(N)

theta = theta[:, np.newaxis]

theta — углы для генерации данных (случайные значения в диапазоне [0;2π]).

*Генерация данных для каждого класса:*

class0 = np.empty((N, col))

class1 = np.empty((N, col))

r = radii0[0] + np.random.rand(N)

r = r[:, np.newaxis]

class0[:, 0] = (r \* np.sin(theta) + cen0[0]).flatten()

r = radii1[0] + np.random.rand(N)

r = r[:, np.newaxis]

class1[:, 0] = (r \* np.sin(theta) + cen1[0]).flatten()

Данные для признака с индексом 0 генерируются с использованием функции sin.

*Повторение для остальных признаков:*

for i in range(1, col):

r = radii0[i] + np.random.rand(N)

r = r[:, np.newaxis]

class0[:, i] = (r \* np.cos(theta) + cen0[i]).flatten()

r = radii1[i] + np.random.rand(N)

r = r[:, np.newaxis]

class1[:, i] = (r \* np.cos(theta) + cen1[i]).flatten()

Данные для остальных признаков генерируются с использованием функции cos.

При выводе на график scatter признака с индексом 0 и любого другого признака, будет получена окружность.

*Создание меток классов и объединение данных:*

Y1 = np.ones((N, 1), dtype=bool)

Y0 = np.zeros((N, 1), dtype=bool)

X = np.vstack((class0, class1))

Y = np.vstack((Y0, Y1)).ravel()

*Перемешивание данных:*

rng = np.random.default\_rng()

arr = np.arange(2\*N)

rng.shuffle(arr)

X = X[arr]

Y = Y[arr]

**Полученные графики**

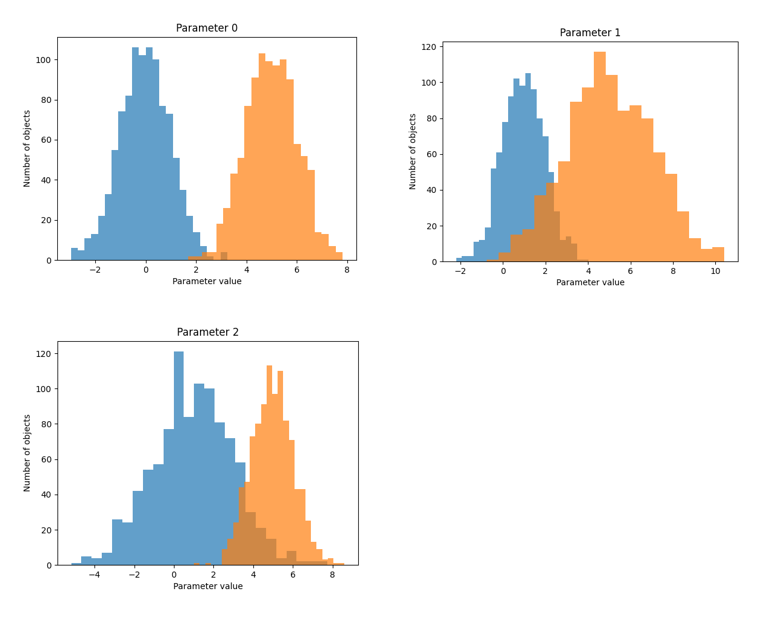


Рис. 1. Гистограммы для данных с нормальным распределением (2 класса)

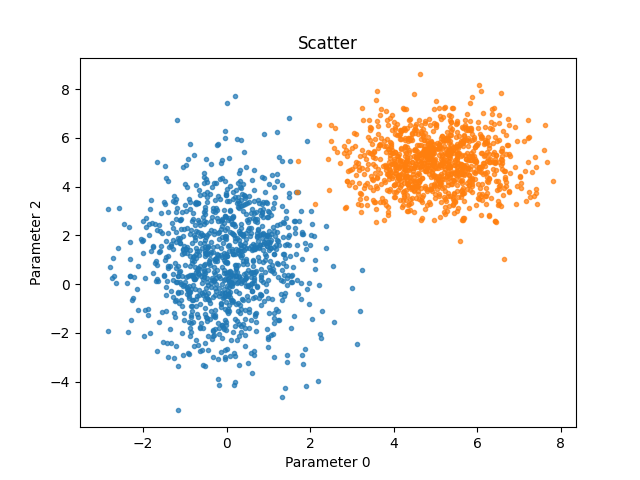


Рис. 2. Точечная диаграмма для данных с нормальным распределением для параметра 0 и параметра 2 (2 класса)

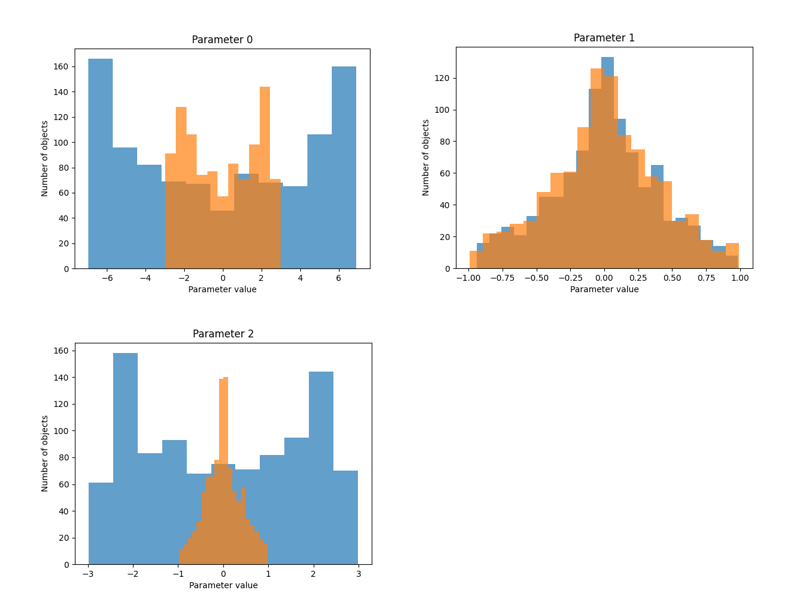


Рис. 3. Гистограммы для данных с нелинейным распределением (2 класса)

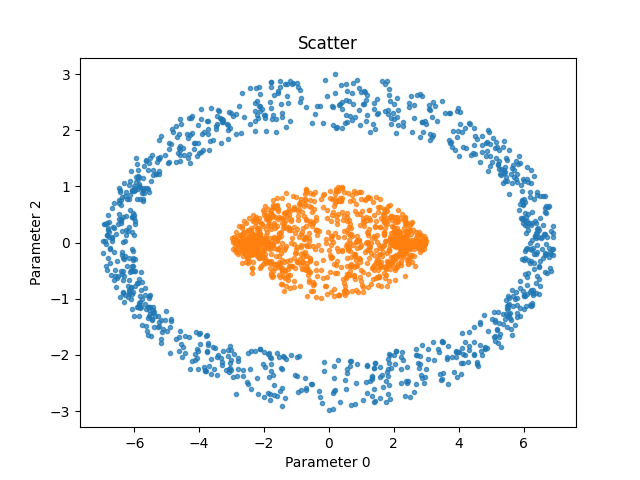


Рис. 4. Точечная диаграмма для данных с нелинейным распределением для параметра 0 и параметра 2 (2 класса)

**Выводы**

В ходе работы были успешно сгенерированы два набора данных: линейный (с использованием нормального распределения) и нелинейный (с использованием тригонометрических функций). Построены гистограммы распределения для каждого признака, что позволило наглядно оценить распределение данных в каждом классе. Построена скатеррограмма (scatter plot) для выбранных признаков, что помогло визуализировать взаимосвязь между признаками и разделимость классов. Данные были разделены на обучающую и тестовую выборки в соотношении 70% на обучение и 30% на тестирование. Это стандартный подход, который позволяет оценить качество модели на независимых данных.