|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 3 |

**Название:** Интеллектуальный анализ данных об объекте мониторинга

**Дисциплина:** Дистанционный мониторинг сложных систем и процессов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ6-12М |  |  | С.В. Астахов | |
|  | (Группа) |  | (Подпись, дата) | | (И.О. Фамилия) |
|  |  |  |  | |  |
| Преподаватель |  |  |  | | Ю.А. Вишневская |
|  |  |  | (Подпись, дата) | | (И.О. Фамилия) |

Москва, 2023

**Введение**

**Цель работы:** разработка и исследование алгоритма принятия решения в условиях неопределенности.

**Задание:** при выполнении лабораторной работы необходимо формализовать задачу принятия решения по распознаванию текущего образа в системе мониторинга, разработать алгоритм распознавания и классификации образа, реализовать интеллектуальный подход при принятии решения в условиях неопределенности. Затем необходимо составить отчет по лабораторной работе с полученными результатами.

**Ход выполнения**

**Исходные данные:** набор эталонов и текущий (распознаваемый) вектор. В качестве эталонов выбрать три образа, например, группу из трех символов (рисунок 1).

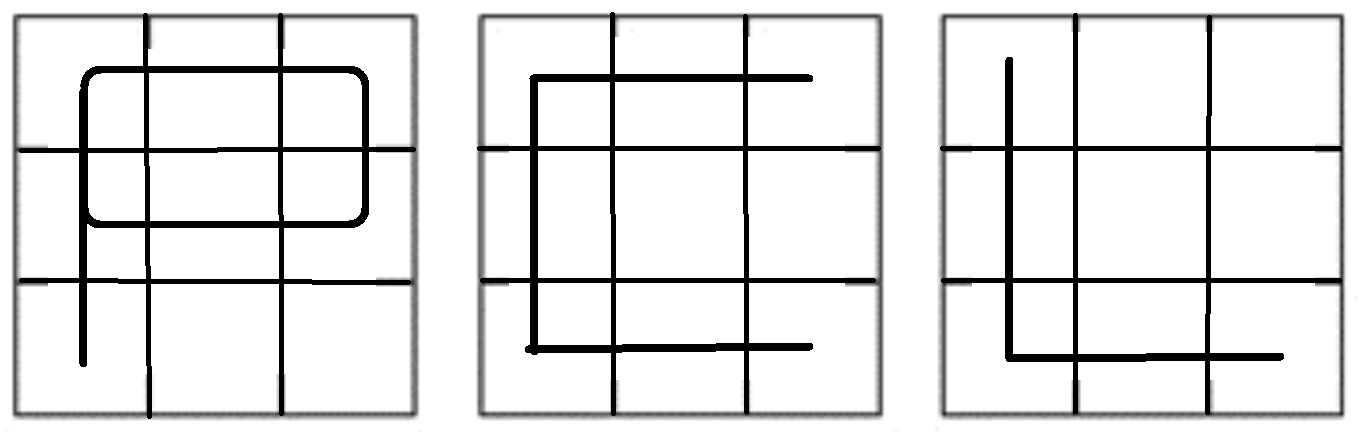


Рисунок 1 — эталонные символы

Здесь эталонные символы вписаны в матрицу 3×3. Таким образом, вектор признаков имеет длину, равную 9. В качестве тестового символа был выбран символ, представленный на рисунке 2.

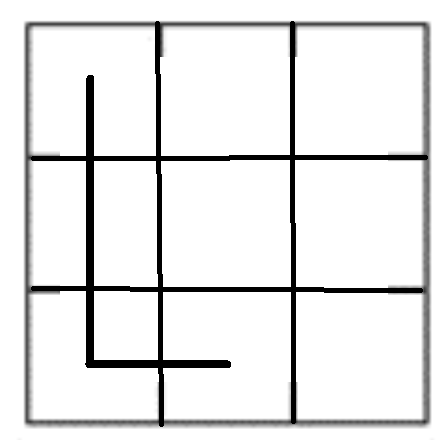


Рисунок 2 — тестовый символ

Для удобства изложения, основной ход выполнения работы описан в комментариях к листингам. Консольный вывод программы приведен в комментариях с началом “(stdout)”. Решение выполнено на языке python. В лиистинге 1 приведен исходный код перевода векторов признаков в десятичную систему

Листинг 1 — перевод векторов признаков в десятичную систему

|  |
| --- |
| import numpy as np  from scipy.integrate import odeint  import matplotlib.pyplot as plt  # вектора признаков  # 01 - вертикальная линия  # 10 - горизонтальная линия  # 11 - пересечение  P = [      11, 10, 11,      11, 10, 11,       1,  0,  0  ]  C = [      11, 10, 10,       1,  0,  0,      11, 10, 10  ]  L = [       1,  0,  0,       1,  0,  0,      11, 10,  10  ]  # перевод кодов признаков в десятичную систему  v1 = list(map(lambda x: int(str(x), 2), P))  v2 = list(map(lambda x: int(str(x), 2), C))  v3 = list(map(lambda x: int(str(x), 2), L))  print("вектора в десятичной системе:\n\n" + str(v1) + "\n" + str(v2) + "\n" + str(v3))  # (stdout) вектора в десятичной системе:  #  # (stdout) [3, 2, 3, 3, 2, 3, 1, 0, 0]  # (stdout) [3, 2, 2, 1, 0, 0, 3, 2, 2]  # (stdout) [1, 0, 0, 1, 0, 0, 3, 2, 2] |

В листинге 2 демонстрируется процесс централизации и нормализации векторов

Листинг 2 — централизация и нормализация

|  |
| --- |
| # централизация векторов  avg1 = sum(v1) / len(v1)  avg2 = sum(v2) / len(v2)  avg3 = sum(v3) / len(v3)  v1 = list(map(lambda x: x - avg1, v1))  v2 = list(map(lambda x: x - avg2, v2))  v3 = list(map(lambda x: x - avg3, v3))  # нормализация векторов  ln1 = sum(list(map(lambda x: x\*x, v1)))  v1 = list(map(lambda x: x / (ln1 \*\* 0.5), v1))  ln2 = sum(list(map(lambda x: x\*x, v2)))  v2 = list(map(lambda x: x / (ln2 \*\* 0.5), v2))  ln3 = sum(list(map(lambda x: x\*x, v3)))  v3 = list(map(lambda x: x / (ln3 \*\* 0.5), v3))  # проверка  ln1 = sum(list(map(lambda x: x\*x, v1)))  ln2 = sum(list(map(lambda x: x\*x, v2)))  ln3 = sum(list(map(lambda x: x\*x, v3)))  print(f"Длины векторов: {ln1}, {ln2}, {ln3}")  print(f"Сумма компонентов векторов: {sum(v1)}, {sum(v2)}, {sum(v3)}")  print("\nНормализованные вектора:\n\n"+str(v1)+"\n\n"+str(v2)+"\n\n"+str(v3))  ””” (stdout)  Длины векторов: 1.0, 0.9999999999999998, 1.0  Сумма компонентов векторов: 0.0, -2.498001805406602e-16, 0.0  Нормализованные вектора:  [0.30949223029508643, 0.030949223029508657, 0.30949223029508643, 0.30949223029508643, 0.030949223029508657, 0.30949223029508643, -0.24759378423606915, -0.5261367915016469, -0.5261367915016469]  [0.42163702135578385, 0.10540925533894595, 0.10540925533894595, -0.21081851067789198, -0.5270462766947299, -0.5270462766947299, 0.42163702135578385, 0.10540925533894595, 0.10540925533894595]  [0.0, -0.31622776601683794, -0.31622776601683794, 0.0, -0.31622776601683794, -0.31622776601683794, 0.6324555320336759, 0.31622776601683794, 0.31622776601683794]  ””” |

В листинге 3 представлен процесс расчета сопряженных векторов.

Листинг 3 — расчет сопряженных векторов

|  |
| --- |
| vp = np.matrix([v1, v2, v3])  w = vp.T \* vp  a = np.linalg.pinv(w) # расчет обратной матрицы  v11 = a\*np.matrix([v1]).T  v22 = a\*np.matrix([v2]).T  v33 = a\*np.matrix([v3]).T  print("Сопряженные вектора")  print(v11)  print(v22)  print(v33)  ””” (stdout)  Сопряженные вектора  [0.4700369583352917, -0.5267655567550685, 0.0567285984197761, 0.7374717794570949, -0.2593307356332636, 0.3241634195415805, 0.3971001889384358, -0.5997023261519239, -0.5997023261519231]  [0.5639276188562116, 0.5674967810008715, 0.4497144302271059, -0.47826772738438167, -0.4746985652397232, -0.5924809160134885, -0.014276648578637698, -0.010707486433978609, -0.01070748643397911]  [-0.007138324289318284, -1.0279186976619563, -0.5460818081329146, 0.8066306446930624, -0.2141497286795736, 0.2676871608494679, 0.920843833322169, -0.09993654005046874, -0.09993654005046743]  ””” |

В листинге 4 представлен процесс предобработки тестового вектора.

Листинг 4 — предобработка тестового вектора

|  |
| --- |
| # вектор признаков  # 01 - вертикальная линия  # 10 - горизонтальная линия  # 11 - пересечение  # измененный символ "L"  Q = [  1, 0, 0,  1, 0, 0,  11, 10, 0  ]  # перевод в десятичную систему  vq = list(map(lambda x: int(str(x), 2), Q))  print("вектор в десятичной системе:\n\n" + str(vq))  # центрирование  avgq = sum(vq) / len(vq)  vq = list(map(lambda x: x - avgq, vq))  # нормализация  lnq = sum(list(map(lambda x: x\*x, vq)))  vq = list(map(lambda x: x / (lnq \*\* 0.5), vq))  lnq = sum(list(map(lambda x: x\*x, vq)))  print(f"Длина вектора: {lnq}")  print(f"Сумма компонентов вектора: {sum(vq)}")  print("Центрированный и нормализованный вектор:\n\n" + str(vq))  ””” (stdout)  вектор в десятичной системе:  [1, 0, 0, 1, 0, 0, 3, 2, 0]  Длина вектора: 0.9999999999999997  Сумма компонентов вектора: 1.1102230246251565e-16  Центрированный и нормализованный вектор:  [0.07188851546895893, -0.25160980414135625, -0.25160980414135625, 0.07188851546895893, -0.25160980414135625, -0.25160980414135625, 0.7188851546895894, 0.3953868350792742, -0.25160980414135625]  ””” |

В листинге 5 приведен процесс расчета параметров порядка и решения ДУ.

Листинг 5 — расчет параметров порядка и сопоставление с эталонами

|  |
| --- |
| # расчет параметров порядка  zeta1 = np.array(v11) @ np.array(vq)  zeta2 = np.array(v22) @ np.array(vq)  zeta3 = np.array(v33) @ np.array(vq)  print("Параметры порядка: ", zeta1, zeta2, zeta3)  # (stdout)  # Параметры порядка: 0.3880053895530886 0.0069277077372853725 1.0876501147537  p\_arr = [zeta1, zeta2, zeta3]  labels = ["P", "C", "L"]  # описание ДУ  def returns\_dpdt1(p, t):  lamb = 1  B = -2  C = 3  p\_ex = list(set(p\_arr) - set([p\_arr[ii]]))  # print(p\_ex)  dpdt = lamb \* p - B \* p \* sum(list(map(lambda x: x\*2, p\_ex))) - C \* p \* sum(list(map(lambda x: x\*2, p\_arr)))  return dpdt    # точки на оси времени  t = np.linspace(0,5)  for ii in range(3):  p0 = p\_arr[ii] # начальное условие  p = odeint(returns\_dpdt1, p0, t) # решение ДУ  plt.plot(t,p, label="эталон " + labels[ii])    # вывод графика  plt.legend()  plt.xlabel("t")  plt.ylabel("p")  plt.show() |

Графическое решение системы ДУ приведено на рисунке 3.

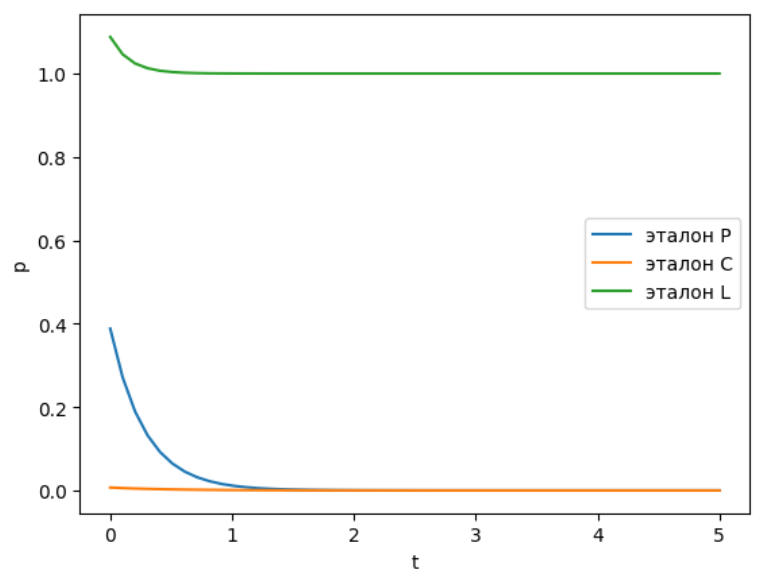


Рисунок 3 — графическое решение системы ДУ

Третий параметр порядка имеет максимальное значение и после динамического изменения системы значение эталона ν3 стремится к 1, а остальные – к 0, следовательно, тестируемый вектор q больше всего похож на эталон ν3 ="L".

**Вывод:** в ходе лабораторной работы была изучена техника принятия решения по распознаванию текущего образа разработан алгоритм распознавания и классификации образа, реализован интеллектуальный подход при принятии решения в условиях неопределенности.

**Контрольные вопросы**

1. Что понимают под процессом принятия решений в системах мониторинга?

2. Почему решения в системах мониторинга принимают в условиях неопределенности? Поясните ответ.

3. Что понимают под неопределенностью информации?

4. Поясните технологию Data Mining, ее цели и задачи?

5. Укажите основные этапы интеллектуального анализа данных;

6. Поясните задачу классификации объектов. Чем она отличается от задачи кластеризации?

7. С какой целью организуют защиту данных в системах мониторинга?