Санкт-Петербургский Политехнический
Университет им. Петра Великого

Институт прикладной математики и механики

Кафедра прикладной математики

Лабораторная работа №1
Расстояние Фреше

4 курс, группа 43631/2

Студент Д. А. Плаксин

Преподаватель Баженов А. Н.

Санкт-Петербург
2019 г.

Содержание

1 Список иллюстраций

2 Постановка задачи

В данной задаче требуется построить ломанные кривые, реализовать вычисление расстояния Фреше между двумя ломанными и найти элементы точки, на которых вычислено это расстояние, с обоснованием точности результата и проверкой единственности.

3 Теория

В ходе решения некоторых математических задач возникает потребность в геометрической оценке свойств полученных множеств. Такой количественной оценкой может служить мера сходства форм областей.

Рассмотрим метрическое пространство с заданной на нём метрикой – (V, d) Стандартный подход к вычислению расстояния Фреше между кривыми – вычисление дискретного расстояния Фреше для ломаных, которые приближают исходные кривые.

Стандартный подход к вычислению расстояния Фреше между кривыми – вычисление дискретного расстояния Фреше для ломаных, которые приближают исходные кривые.

Пусть $P : [0, n] \to V$ ломаная кривая; Q – ломаная кривая. L – сопряжение между двумя кривыми.

Тогда дискретное расстояние Фреше:

$$\delta_{dF}(P, Q) = min\|L\| \tag{1}$$

4 Реализация

Для генерации выборки был использован *Python* 3.7. Использовалась библиотека numpy, графики строились с помощью библиотеки *matplotlib*.

5 Результаты

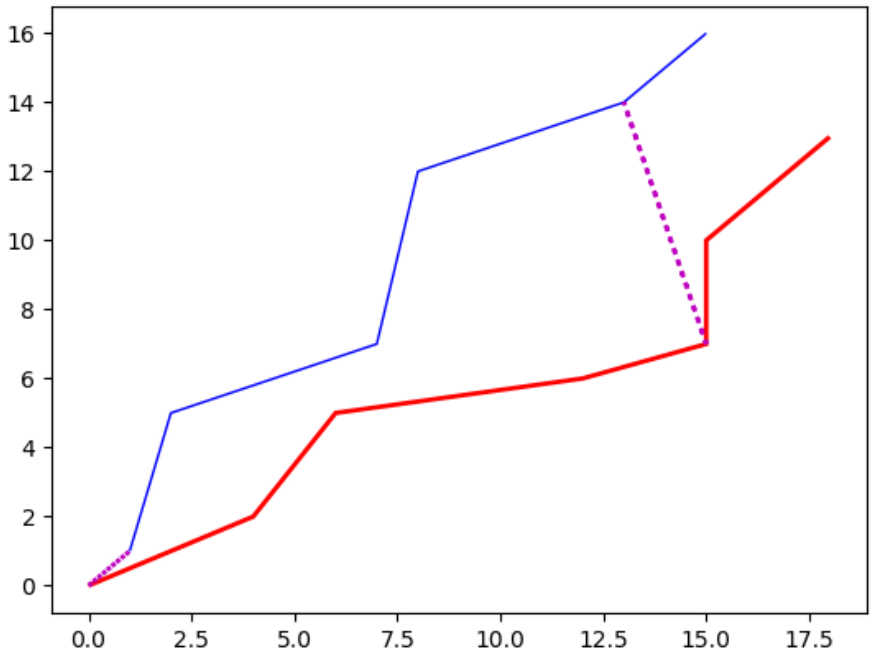
Для кривых:

$$P = [[0, 0], [4, 2], [6, 5], [12, 6], [15, 7], [15, 10], [18, 13]]$$

$$Q = [[1, 1], [2, 5], [7, 7], [8, 12], [13, 14], [15, 16]]$$

Ответ: $\delta_{dF}(P, Q) = 7.280109889280518$ между точками $(15, 7)$ и $(13, 14)$

Рис. 1: Расстояние Фреше для двух кривых



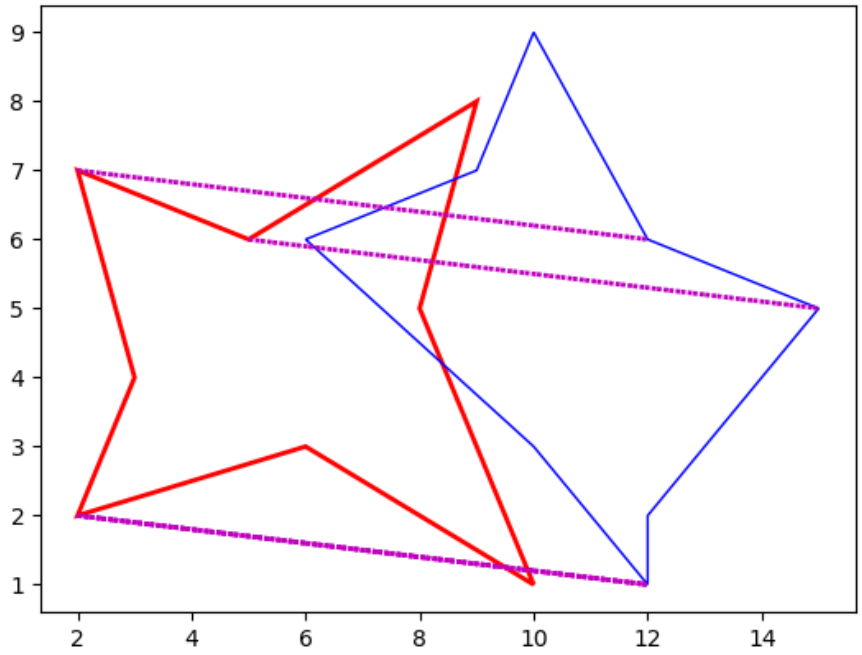
Для двух замкнутых кривых, ограничивающих невыпуклые множества:

$$P = [[2,2],[3,4],[2,7],[5,6],[9,8],[8,5],[10,1],[6,3],[2,2]]$$

$$Q = [[12,1],[10,3],[6,6],[9,7],[10,9],[12,6],[15,5],[13,3],[12,1]]$$

Ответ: $\delta_{dF}(P,Q) = 10.04987562112089$.

Рис. 2: Расстояние Фреше для замкнутых кривых.



6 Обсуждение

6.1 Точность результатов

Библиотека *numpy* оперирует числами с плавающей запятой двойной точности – $float64$, таким образом поддерживается точность до 15 – 17 знака после запятой.

Из-за вычисления нормы в ходе расчётов точность снижается в два раза, так как алгоритм включает в себя вычисление нормы, содержащей операцию перемножения двух чисел $float64$ и составляет 7 знаков после запятой.

6.2 Единственность

Для проверки единственности решения достаточно проверить на каждом шаге вычислений:

$$dist = \delta_{dF}(P[i],Q[j]) = \max\left(\min(c[i-1,j],c[i-1,j-1],c[i,j-1],\delta_{dF}(P[i],Q[j])\right) \quad (2)$$

Тогда значение в $[i,j]$ оказывается равным расстоянию в точках $P[i],Q[j]$. Значит, решение не является единственным и расстояние Фреше можно найти между двумя другими точками.

6.3 Трудоёмкость

Алгоритм вычисления расстояния Фреше является рекурсивным, и наибольшую трудоёмкость имеют операции вычисления следующего выражения, которое часто вычисляется:

$$\max\left(\min\left(d\left(a_{k_{i-1}}, b_{m_j}\right), d(\left(a_{k_{i-1}}, b_{m_{j-1}}\right), d\left(a_{k_i}, b_{m_{j-1}}\right), d\left(a_{k_i}, b_{m_j}\right)\right)\right) \tag{3}$$

7 Список литературы

[1] Модуль numpy - https://physics.susu.ru/vorontsov/language/numpy.html

[2] Модуль matplotlib - https://matplotlib.org/users/index.html

[3] Пособие к Лабораторным работам https://cloud.mail.ru/public/4ra6/5wwqBzMBC/LabPractics.pdf

8 Приложения

Код отчёта: https://github.com/MisterProper9000/computing-complexes-labs/blob/frechet-distance/frechet.tex

Код лабораторной: https://github.com/MisterProper9000/computing-complexes-labs/blob/frechet-distance/frechet.py

```
import numpy as np
import matplotlib.patches as mpatches
import matplotlib.pyplot as plt

answer = -1
answer_ind = []

def checkAnswer(dst, i, j, pred):
    global answer, answer_ind
    if pred == 0: pair_prev = [i-1, j]
    elif pred == 1: pair_prev = [i-1, j-1]
    else: pair_prev = [i, j-1]
    if (answer == dst):
        answer = dst
        answer_ind.append([i, j])
    else:
        answer = dst
        answer_ind.clear()
        answer_ind.append([i, j])
        answer_ind.append(pair_prev)
    return

def _c(ca, i, j, p, q):
    global answer_ind, answer
    if ca[i, j] > -1:
        return ca[i, j]
    elif i == 0 and j == 0:
        ca[i, j] = np.linalg.norm(p[i]-q[j])
        answer = ca[0, 0]
        answer_ind.append([0,0])
    elif i > 0 and j == 0:
        t_dst = np.linalg.norm(p[i] - q[j])
        ca[i, j] = max(_c(ca, i - 1, 0, p, q), t_dst)
        if ca[i - 1, j] == t_dst:
            checkAnswer(t_dst, i, j, 0)

    elif i == 0 and j > 0:
        t_dst = np.linalg.norm(p[i] - q[j])
```

```
        ca[i, j] = max(_c(ca, 0, j - 1, p, q), t_dst)
        if ca[i, j - 1] == t_dst:
            checkAnswer(t_dst, i, j, 2)

    elif i > 0 and j > 0:
        t_dst = np.linalg.norm(p[i] - q[j])
        ca[i, j] = max(
            min(
                _c(ca, i - 1, j, p, q),
                _c(ca, i - 1, j - 1, p, q),
                _c(ca, i, j - 1, p, q)
            ),
            t_dst
        )
        if ca[i - 1, j - 1] == t_dst:
            checkAnswer(t_dst, i, j, 1)
        elif ca[i - 1, j] == t_dst:
            checkAnswer(t_dst, i, j, 0)
        elif ca[i, j - 1] == t_dst:
            checkAnswer(t_dst, i, j, 2)
    else:
        ca[i, j] = float('inf')

    return ca[i, j]


def frechetDist(p, q):
    p = np.array(p, np.float64)
    q = np.array(q, np.float64)

    len_p = len(p)
    len_q = len(q)

    if len_p == 0 or len_q == 0:
        raise ValueError('Input curves are empty.')

    ind1 = len_p - 1
    ind2 = len_q - 1
    ca = (np.ones((len_p, len_q), dtype=np.float64) * -1)
    dist = _c(ca, len_p - 1, len_q - 1, p, q)
    while (True):
        if (ind1 > 0 and ind2 > 0 and dist == ca[ind1 - 1, ind2 - 1]):
            ind1 = ind1 - 1
            ind2 = ind2 - 1
        if (ind1 > 0 and dist == ca[ind1 - 1, ind2]):
            ind1 = ind1 - 1
        elif (ind2 > 0 and dist == ca[ind1, ind2 - 1]):
            ind2 = ind2 - 1
        else:
            break
    return dist, ind1, ind2

def draw(axes, P, Q, ind1, ind2, cls, addit_ind):
    polygon_1 = mpatches.Polygon(P,
                                 fill=False,
                                 closed=cls, color='red', linewidth=2)
    axes.add_patch(polygon_1)
    polygon_2 = mpatches.Polygon(Q,
                                 fill=False,
                                 closed=cls, color='blue')
    axes.add_patch(polygon_2)
    line_1 = mpatches.Polygon([P[ind1], Q[ind2]], color='m', linestyle=':',
    linewidth=2)
    axes.add_patch(line_1)
    for pair in addit_ind:
        line = mpatches.Polygon([P[pair[0]], Q[pair[1]]], color='m',
```

```
    linestyle=':', linewidth=2)
104         axes.add_patch(line)
105
106 ax = plt.gca()
107 ax.set_xlabel('x')
108 ax.set_ylabel('y')
109 ax.set_title('Fresche distance')
110 ax.grid(True)
111 plt.legend(handles=[
112     mpatches.Patch(color='red', label='P'),
113     mpatches.Patch(color='blue', label='Q'),
114     mpatches.Patch(color='magenta', label='dist')
115 ], loc='best')
116
117
118 plt.cla()
119 #P = [[2, 3], [3, 4], [2, 7], [5, 6], [9, 8], [6, 18], [10, 1], [6, 3]]
120 #Q = [[12, 1], [10, 3], [6, 6], [9, 7], [10, 5], [12, 6], [15, 5], [13, 3]]
121
122 #P = [[0, 0],[4, 2],[6, 5],[12, 6],[15, 7],[15, 10],[18, 13]]
123 #Q = [[1, 1],[2, 5],[7, 7],[8, 12],[13, 14],[15, 16]]
124
125 #P=[[1,1], [2,1], [2,2]]
126 #Q=[[2,2], [0,1], [2,4]]
127
128 P = [[2, 2], [3, 4], [2, 7], [5, 6], [9, 8], [8, 5], [10, 1], [6, 3], [2, 2]]
129 Q = [[12, 1], [10, 3], [6, 6], [9, 7], [10, 9], [12, 6], [15, 5], [12, 2],
    [12, 1]]
130 closed = True
131 d, ind1, ind2 = frechetDist(P, Q)
132 print(d)
133 print(ind1, ind2)
134 print(answer_ind, answer)
135 draw(ax, P, Q, ind1, ind2, closed, answer_ind)
136 plt.plot()
137 plt.show()
138
139
140 #P=[[1,1], [2,1], [2,2]]
141 #Q=[[2,2], [0,1], [2,4]]
142
143 #P=[[1,1], [2,1], [2,2]]
144 #Q=[[1,1], [2,1], [2,2]]
145
146 #P = [[0, 0],[4, 2],[6, 5],[12, 6],[15, 7],[15, 10],[18, 13]]
147 #Q = [[1, 1],[2, 5],[7, 7],[8, 12],[13, 14],[15, 16],[15,16]]
```