DP マッチングによる単語音声認識

2024 年度開講 認識工学レポート

千葉工業大学 先進工学部 未来ロボティクス学科

22C1704 鷲尾 優作

2024 年 6 月 19 日

1　目的

本稿では，DP マッチングを用いた単語音声認識の手法について実験を行い，その正答率を評価することで，単語音声認識における DP マッチングの有効性を検証する．

2　実験

実験には，単語音声認識のための音声データセットを用いる．音声入力から音声分析までの過程は既に終了しているものとし，あらかじめ用意された音声データセットを用いて，DP マッチングによる単語音声認識を行う．

2.1　DP マッチング

DP マッチングは，動的計画法を用いて，2 つの系列データ間の類似度を計算する手法である．2 つの系列データを比較し，最も類似度の高い経路を探索することで，系列データ間の類似度を計算する．類似度は，探索した経路の経路長が短いほど高くなる．本実験では，音声データのメルケプストラム特徴量を用いて，DP マッチングによる単語音声認識を行う．

2.2　データの内容

音声データセットは，2 名が 2 回ずつ発声した 100 単語のデータで構成されている．内容は，100 単語の地名単語データベースであり，各単語は日本国内の地名を表すものである．話者 2 名がそれぞれ 2 回ずつ発声した計 400 単語のメルケプストラム特徴量が記録されており，話者 1 の 1 回目「city011」話者 1 の 2 回目「city012」話者 2 の 1 回目「city021」話者 2 の 2 回目「city022」の 4 つのデータが含まれている．

2.3　実験手順

本実験では，1 音声データに対し，もうひとつの 100 単語を順番に比較し，DP マッチングにより最も類似度の高い単語を認識する．また，比較は同一話者による音声データ同士，別話者による音声データ同士の 2 通りで行い，計 6 通りの組み合わせで正答率を比較する．

比較には Listing 1 に示す Python スクリプトを用いた．

2.4　実験結果

実験結果を表 1 に示す．表中の「同一話者」は，同一話者による音声データ同士の比較結果を示し，「別話者」は別話者による音声データ同士の比較結果を示す．同一の比較にあたる部分は空欄としている．

表 1: DP マッチングによる単語音声認識の正答率

比較した話者	話者 1(1 回目)	話者 1(2 回目)	話者 2(1 回目)
同一話者	99%		99%
別話者 (1 回目)	90%	92%	
別話者 (2 回目)	84%	86%	

3 考察

表 1 より，同一話者による音声データ同士の比較では，2 つのデータでそれぞれ 99% と最も高い正答率を示した．一方，別話者による音声データ同士の比較では，正答率が 10% 程度低下している．これは，同一話者による音声データ同士の方が，発声の特徴が類似しているため，DP マッチングによる認識が容易であること，別話者の場合，発声の特徴が異なるため，DP マッチングによる認識が難しくなることを示すと考えられる．

検証のため，DP マッチングにより得られた経路をバックトラックし，認識結果を確認した．図 1 に，話者 1 の 1 回目「city011」と話者 1 の 2 回目「city012」の音声データを比較した結果，図 2 に，話者 1 の 1 回目「city011」と話者 2 の 1 回目「city021」の音声データを比較した結果を示す．

それぞれ，図中の赤い線が DP マッチングにより得られた経路を表し，100 単語の最も最初のデータである「AZABU」を認識した結果を示している．

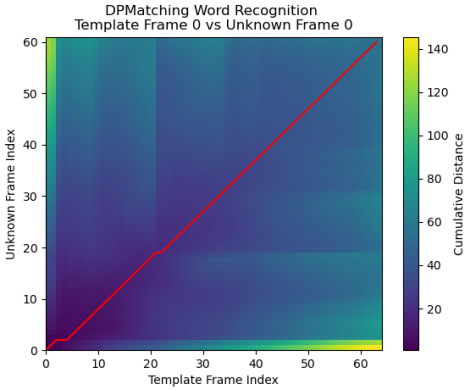

図 1: 話者 1 の 1 回目「city011」と話者 1 の 2 回目「city012」の音声データを比較した結果

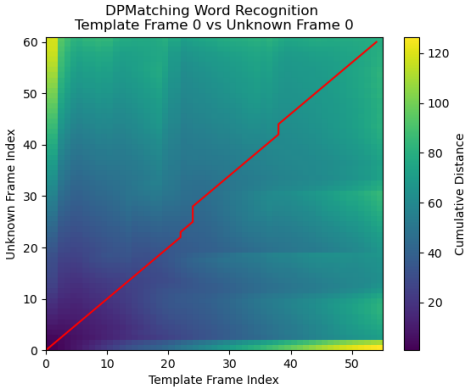

図 2: 話者 1 の 1 回目「city011」と話者 2 の 1 回目「city021」の音声データを比較した結果

両画像とも，赤い線が左上から右下に向かって伸びていることから，DP マッチングにより正しく認識されたことが確認できる．この経路の距離が短いほど，認識精度が高くなるが，図 2 はほぼ直線の図 1 に比べてギザギザした経路となっていることから，認識精度が低いことが示されている．

このことは，表 1 の結果と一致しており，別話者による音声データ同士の比較では，正答率が低下することが確認された．

4　結論

本稿では，DP マッチングを用いた単語音声認識の手法について実験を行い，その正答率を評価した．実験結果より，同一話者による音声データ同士の比較では，DP マッチングによる認識が容易であることが示された．一方，別話者による音声データ同士の比較では，正答率が低下することが確認された．

Listing 1: main.py

```
import numpy as np
import os
import matplotlib.pyplot as plt
import time

maxs = 254
dim = 15
file_num = 100
file_output_path = "output.txt"
output_folder = "output/"
temp_f_n = 11
unkn_f_n = 21


class McepData:
    def __init__(self):
        self.name = ""
        self.onso = ""
        self.flame = 0
        self.mcepdata = np.zeros((maxs, dim))


def read_mcep_file(filename):
    with open(filename, 'r') as file:
        mcepdata = McepData()
        mcepdata.name = file.readline().strip()
        mcepdata.onso = file.readline().strip()
        mcepdata.flame = int(file.readline().strip())
        for i in range(mcepdata.flame):
            mcepdata.mcepdata[i] = list(
                map(float, file.readline().strip().split()))
    return mcepdata


def calk_dis(template_file, unknown_file):
    d = np.zeros((maxs, maxs))
    g = np.zeros((maxs, maxs))

    for i in range(template_file.flame):
        for j in range(unknown_file.flame):
            d[i, j] = np.sum((template_file.mcepdata[i] -
                              unknown_file.mcepdata[j]) ** 2)
            d[i, j] = np.sqrt(d[i, j])

```

```
45      g[0, 0] = d[0, 0]
46
47      for i in range(1, template_file.flame):
48          g[i, 0] = g[i - 1, 0] + d[i, 0]
49      for j in range(1, unknown_file.flame):
50          g[0, j] = g[0, j - 1] + d[0, j]
51
52      for i in range(1, template_file.flame):
53          for j in range(1, unknown_file.flame):
54              a = g[i, j - 1] + d[i, j]
55              b = g[i - 1, j - 1] + 2 * d[i, j]
56              c = g[i - 1, j] + d[i, j]
57              g[i, j] = min(a, b, c)
58
59      path = [(template_file.flame - 1, unknown_file.flame - 1)]
60      i, j = template_file.flame - 1, unknown_file.flame - 1
61      while i > 0 or j > 0:
62          if i == 0:
63              j -= 1
64          elif j == 0:
65              i -= 1
66          else:
67              steps = [(i - 1, j), (i, j - 1), (i - 1, j - 1)]
68              costs = [g[step] for step in steps]
69              min_step = steps[np.argmin(costs)]
70              i, j = min_step
71          path.append((i, j))
72      path.reverse()
73
74      return g, g[template_file.flame - 1, unknown_file.flame - 1] / (template_file.flame
            + unknown_file.flame), path
75
76
77  def save_dtw_plot(g, template_idx, unknown_idx, path, template_flame, unknown_flame):
78      plt.figure()
79      plt.imshow(g[:template_flame, :unknown_flame], origin='lower', cmap='viridis',
80                 interpolation='none', extent=[0, unknown_flame, 0, template_flame])
81      plt.colorbar(label='Cumulative Distance')
82      plt.title("DPMatching Word Recognition \nTemplate Frame {} vs Unknown Frame {}".
            format(
83          template_idx, unknown_idx))
84      plt.xlabel('Template Frame Index')
85      plt.ylabel('Unknown Frame Index')
86
87      path = np.array(path)
88
```

```
89      plt.plot(path[:, 1], path[:, 0], 'r')
90      plt.axis('tight')
91
92      os.makedirs(output_folder, exist_ok=True)
93      plt.savefig(os.path.join(output_folder, "city{0:03d}_{1:03d}_vs_city{2:03d}_{3:03d}.
            png".format(
94          temp_f_n, template_idx, unkn_f_n, unknown_idx)))
95      print(os.path.join(output_folder, "city{0:03d}_{1:03d}_vs_city{2:03d}_{3:03d}.png".
            format(
96          temp_f_n, template_idx, unkn_f_n, unknown_idx)))
97      plt.close()
98
99
100 def main():
101     count = 0
102     first_comparison_done = False
103     start = time.time()
104
105     for h0 in range(file_num):
106         temp_filename = "city_mcepdata/city{0:03d}/city{0:03d}_{1:03d}.txt".format(
107             temp_f_n, h0 + 1)
108         template_file = read_mcep_file(temp_filename)
109
110         word_dis = np.zeros(file_num)
111
112         for h in range(file_num):
113             miti_filename = "city_mcepdata/city{0:03d}/city{0:03d}_{1:03d}.txt".format(
114                 unkn_f_n, h + 1)
115             unknown_file = read_mcep_file(miti_filename)
116
117             g, word_dis[h], path = calk_dis(template_file, unknown_file)
118
119             if not first_comparison_done:
120                 save_dtw_plot(g, h0, h, path, template_file.flame,
121                               unknown_file.flame)
122                 first_comparison_done = True
123
124             elapsed_time = time.time() - start
125             print("elapsed_time:{0:.3f},␣h0:{1},␣h:{2},␣word_dis:{3:.3f}".format(
126                 elapsed_time, h0, h, word_dis[h]))
127
128         word_dis_min = np.min(word_dis)
129         num_match_fname = np.argmin(word_dis)
130
131         if num_match_fname == h0:
132             print("Matching")
```

```
133             print("city{0:03d}_{1:03d}".format(temp_f_n, h0 + 1))
134             print("city{0:03d}_{1:03d}".format(unkn_f_n, num_match_fname + 1))
135             print("word␣distance␣:␣{}".format(word_dis_min))
136             count += 1
137             print(count)
138
139         if num_match_fname != h0:
140             print("NOT␣Matching")
141             print("city{0:03d}_{1:03d}".format(temp_f_n, h0 + 1))
142             print("city{0:03d}_{1:03d}".format(unkn_f_n, num_match_fname + 1))
143             print("word␣distance␣:␣{}".format(word_dis_min))
144
145     with open(file_output_path, 'a') as fp_output:
146         fp_output.write("正答率{}%\n".format(count))
147     print("\ファイルを作成しました。n")
148     print("正答率␣{}%␣".format(count))
149
150
151 if __name__ == "__main__":
152     main()
```

参考文献

[1] 大川茂樹. 2024 認識工学_hmm.pdf, 2024.