認識工学大レポート課題

19C1123 横尾陸

2021 年 6 月 17 日

1 目的

DP マッチングのアルゴリズムを利用し，単語音声認識実験を行い，100 単語中の認識率を調査する．

2 方法

DP マッチングのアルゴリズムのプログラムを実装し，単語の認識率を調べる．今回自作でのプログラムを試みたが実装するまで行かなかった.(1) 19C1012 井上叡さんのコード (2) を用いて実験を行った.

Listing 1 作成したプログラム

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <iomanip>
#include <cmath>

using namespace std;
double local_distance(std::vector<double> fream_i, std::vector<double> fream_j);
double min3(double x, double y, double z);

int dpmaching(std::vector<std::vector<std::vector<double>>> &anser, std::vector<std::
    vector<double>> &detect){
  cout << "stra\n";
  double tmp=0, kai=0,r=0,m=0, l=0, min=0;
  int flag=0, flag2=0;
  std::vector<double> zero(15,0);
  std::vector<double> ans(100);
  for(int i=0;i < 100;i++){
    std::vector<std::vector<double>> total(anser.at(i).size(), std::vector<double>(
        detect.size()));
    cout << "for\n";
    for(int j=0;j<anser.at(i).size();j++){
      for(int k=0;k<detect.size();k++){
```

```
23          if(j==0 && k==0){
24            total.at(0).at(0) = local_distance(zero,zero);
25          }else if(j>0 && k-1 < 0){
26            total.at(j).at(0) = total.at(j-1).at(k) + local_distance(anser.at(i).at(j),
                  zero);
27          }else if(k>0 && j-1 < 0){
28            total.at(0).at(k) = total.at(0).at(k-1) + local_distance(zero, detect.at(k))
                  ;
29          }else{
30            r = total.at(j).at(k-1) + local_distance(anser.at(i).at(j), detect.at(k));
31            m = total.at(j-1).at(k-1) + 2 * local_distance(anser.at(i).at(j), detect.at(
                  k));
32            l = total.at(j-1).at(k) + local_distance(anser.at(i).at(j), detect.at(k));
33            total.at(j).at(k) = min3(r, m, l);
34            }
35        }
36      }
37      ans.at(i) = total.at(anser.at(i).size()).at(detect.size()) / (double)(anser.at(i).
            size() + detect.size());
38      tmp = ans.at(i);
39      if(i==0) {
40        min = ans.at(i);
41        flag = i;
42      }else if(min > tmp){
43        min = tmp;
44        flag = i;
45      }
46    }
47    return flag;
48  }
49
50  double local_distance(std::vector<double> fream_i, std::vector<double> fream_j){
51    double ans=0, tmp=0, kai=0;
52    for(int i=0;i<15;i++){
53      tmp = fream_i.at(i)-fream_j.at(i);
54      ans += tmp*tmp;
55    }
56    return sqrt(ans);
57  }
58
59  double min3(double x, double y, double z)
60  {
61    double min = x;
62
63    if (y < min) min = y;
64    if (z < min) min = z;
```

```
65   return (min);
66 }
67
68
69 int main(){
70   int flag=0, flag2=0;
71   std::vector<std::vector<std::vector<double>>> templeat_data(100), detect_data(100);
72   double ans=0;
73   std::vector<string> str1(3), str2(3);
74   for(int I=0;I<100;I++){
75     std::vector<string> info(3),info2(3),dev(4);
76     stringstream ss,ss2;
77     stringstream stod, stod2;
78     double fream,fream2;
79     std::string filename, filename2;
80     ss << "city011/" << "city011_" << std::setw(3) << std::setfill('0') << I+1 << ".
           txt";
81     ss2 << "city012/" << "city012_" << std::setw(3) << std::setfill('0') << I+1 << ".
           txt";
82     filename = ss.str();
83     filename2 = ss2.str();
84     std::ifstream fin(filename);
85     std::ifstream fin2(filename2);
86     for(int i=0;i<3;i++){
87       fin >> info.at(i);
88       fin2 >> info2.at(i);
89     }
90     stod << info.at(2);
91     stod >> fream;
92     stod2 << info2.at(2);
93     stod2 >> fream2;
94     templeat_data.at(I).resize(fream);
95     detect_data.at(I).resize(fream2);
96     fin.close();
97     fin2.close();
98     fin.open(filename);
99     fin2.open(filename2);
100    for(int i=0;i<fream;i++){
101      for(int j=0;j<15;j++){
102        stringstream ss;
103        double tem;
104        if(i==0){
105          fin >> dev.at(0);
106          fin >> dev.at(1);
107          fin >> dev.at(2);
108        }
```

```
109         vector<string> str(1);
110         fin >> str.at(0);
111         ss << str.at(0);
112         ss >> tem;
113         templeat_data.at(I).at(i).push_back(tem);
114       }
115     }
116     for(int i=0;i<fream2;i++){
117       for(int j=0;j<15;j++){
118         stringstream ss;
119         double tem;
120         if(i==0){
121           fin2 >> dev.at(0);
122           fin2 >> dev.at(1);
123           fin2 >> dev.at(2);
124         }
125         vector<string> str(1);
126         fin2 >> str.at(0);
127         ss << str.at(0);
128         ss >> tem;
129         detect_data.at(I).at(i).push_back(tem);
130       }
131     }
132     cout << I << endl;
133   }
134   for(int i=0;i<100;i++){
135     flag = dpmaching(templeat_data, detect_data.at(i));
136     if(flag == i) flag2++;
137   }
138   cout << flag2 << endl;
139   return 0;
140 }
```

Listing 2　井上さんのプログラム

```
1  #include <fstream>
2  #include <string>
3  #include <vector>
4  #include <iostream>
5  #include <experimental/filesystem>
6  #include <cmath>
7  #include <algorithm>
8  using namespace std;
9  class file_input
10 {
11 public:
12     file_input();
```

```
13      std::ifstream ifs;
14      std::vector<std::string> cities = {"city011", "city012", "city021", "city022"};
15      const std::string root = "/home/riku/GIT/5S/Numerical_Analysis2/DPmaching/";
16      std::vector<std::vector<std::string>> filenames;
17      using data_t = std::vector<std::vector<std::vector<std::vector<float>>>>;
18      data_t data; フォルダファイルが何番目か行列//[][][][]
19      static const int dimension = 15;
20      data_t getfiledatas();
21      float local_distance(const std::vector<float> &frame_i, const std::vector<float> &
            frame_j);
22      int dpMatching(int tmpfolder, int tmpfile_num, int folder);
23  };
24  file_input::file_input()
25  {
26      filenames.resize(4);
27      data.resize(4);
28      for (auto &a : data)
29      {
30          a.resize(100);
31      }
32  }
33  file_input::data_t file_input::getfiledatas()
34  {
35      int city_num = 0;
36      for (const auto &city : cities)
37      {
38          std::string now = root + city;
39          for (const std::experimental::filesystem::directory_entry &i : std::experimental
                ::filesystem::directory_iterator(now))
40          {
41              filenames[city_num].push_back(i.path().filename().string());
42          }
43          std::sort(filenames[city_num].begin(), filenames[city_num].end());
44          int file_count = 0;
45          for (const auto &name : filenames[city_num])
46          {
47              ifs.open(now + "/" + name);
48              std::string s;
49              ifs >> s;
50              ifs >> s;
51              int rows = 0;
52              ifs >> rows;
53              data[city_num][file_count].resize(rows);
54              for (int i = 0; i < rows; ++i)
55              {
56                  float tmp;
```

```
57                 for (int j = 0; j < dimension; ++j)
58                 {
59                     ifs >> tmp;
60                     data[city_num][file_count][i].push_back(tmp);
61                 }
62             }
63             ifs.close();
64             file_count++;
65         }
66         city_num++;
67     }
68     return this->data;
69 }
70 int file_input::dpMatching(int tmpfolder, int tmpfile_num, int target_folder)
71 {
72     float min_distance = 1e9;
73     auto template_data = data[tmpfolder][tmpfile_num];
74     int voice_num = 0;
75     int shortest = 0;
76     for (const auto &target : data[target_folder])
77     {
78         int max_i = template_data.size();
79         int max_j = target.size();
80         std::vector<std::vector<double>> result(template_data.size(), std::vector<double
               >(target.size(), 1e8));
81         result[0][0] = local_distance(template_data[0], target[0]);
82         for (int tmp_i = 0; tmp_i < max_i; ++tmp_i)
83         {
84             for (int target_j = 0; target_j < max_j; ++target_j)
85             {
86                 if (tmp_i == 0 && target_j == 0)
87                     continue;
88                 double min_num = 1e10;
89                 if (tmp_i - 1 >= 0)
90                     min_num = std::min(min_num, local_distance(template_data[tmp_i],
                           target[target_j]) + result[tmp_i - 1][target_j]);
91                 if (target_j - 1 >= 0)
92                     min_num = std::min(min_num, local_distance(template_data[tmp_i],
                           target[target_j]) + result[tmp_i][target_j - 1]);
93                 if ((tmp_i - 1 >= 0) && (target_j - 1 >= 0))
94                     min_num = std::min(min_num, 2 * local_distance(template_data[tmp_i],
                            target[target_j]) + result[tmp_i - 1][target_j - 1]);
95                 result[tmp_i][target_j] = min_num;
96             }
97         }
98         if (min_distance >= result[max_i - 1][max_j - 1] / (max_i + max_j))
```

```
99          {
100             min_distance = result[max_i - 1][max_j - 1] / (max_i + max_j);
101             shortest = voice_num;
102         }
103         voice_num++;
104     }
105     if(shortest == tmpfile_num)
106         return 1;
107     else
108         return 0;
109 }
110 float file_input::local_distance(const std::vector<float> &frame_i, const std::vector<
        float> &frame_j)
111 {
112     float result_distance = 0;
113     for (int i = 0; i < dimension; ++i)
114     {
115         result_distance += (frame_i[i] - frame_j[i]) * (frame_i[i] - frame_j[i]);
116     }
117     return std::sqrt(result_distance);
118 }
119 int main(int argc, char const *argv[])
120 {
121     file_input files;
122     files.getfiledatas();
123     for (int i = 0; i < 4; i++)
124     {
125         for (int j = 0; j < 4; j++)
126         {
127             int n = 0;
128             if (i == j)
129             {
130                 continue;
131             }
132             for (int k = 0; k < 100; ++k)
133             {
134                 n += files.dpMatching(i,k,j);
135             }
136             std::cout << "template::" << files.cities[i] << " compared::" << files.
                    cities[j] << std::endl;
137             std::cout << "result:: " << n << "%"<< std::endl;
138         }
139     }
140     return 0;
141 }
```

3　結果

DPmachig した結果を表 1, 表 2 に示す．表 1 は斜めの移動時に 2 倍した音声認識率で表 2 は斜めの移動時 1 倍した音声認識率である．

表 1　=

斜め遷移の重みを 2 倍にしたときの音声認識率

template\input	city011	city012	city021	city022
city011	x	99%	90%	84%
city012	100%	x	92%	86%
city021	83%	91%	x	99%
city022	86%	94%	100%	x

表 2　=

斜め遷移の重みを 1 倍にしたときの音声認識率

template\input	city011	city012	city021	city022
city011	x	99%	95%	94%
city012	100%	x	96%	98%
city021	92%	99%	x	100%
city022	95%	98%	100%	x

結果を見ると高い確率で認識できているのがわかる．同一話者同士だと 100% が出るなど精度の高さがわかる．斜め遷移を 1 倍に変更すると認識精度があがったのがわかる．

4　考察

斜め移動を 1 倍にしたときに認識精度が上がったのは授業や実際に見たとおりわかるが斜め移動は移動距離が多いので縦横の移動と同じ倍率だと斜めが有利になり認識精度が上がったと考える．