新人課題

津嶋 佑旗

2019 年 4 月 19 日

1 配列情報処理

1.1 ヒト 21 番染色体の先頭コンティグの DNA 配列を秋山研サーバから取得し，A/T/G/C の各塩基数を出力せよ．

以下のように実行する．実行環境は Python2 で，ghostgw 上での動作を確認している．

```
1    python 1_1.py /mnt/fs/ohue/newcomer/NT_113952.1.fasta
```

ソースコード 1　実行方法

プログラム本体は以下の通りである．fasta ファイルの塩基配列について，行ごとに Python の機能で各文字の出現回数をカウントする．

```
1  #2019/04/11
2  #Yuki Tsushima
3  #tsushima@bi.c.titech.ac.jp
4  #For Python 2.7
5
6  import sys
7
8  #file open
9  args = sys.argv
10 if len(args) != 2:
11     sys.stderr.write('Usage: python 1_1.py ./NT_113952.1.fasta\n')
12     sys.exit()
13 path = args[1]
14 try:
15     file = open(path)
16 except:
17     sys.stderr.write('Error in opening file.\n')
18     sys.exit()
19 lines = file.readlines()
20 length = len(lines)
21
22 #counter(num of A/T/G/C)
23 noA = 0
24 noT = 0
25 noG = 0
26 noC = 0
27 #Process with each line
28 #Skip Header
29 for l in range(1, length):
30     noA = noA + lines[l].count('A')
31     noT = noT + lines[l].count('T')
32     noG = noG + lines[l].count('G')
33     noC = noC + lines[l].count('C')
34 print("A:"+str(noA))
35 print("T:"+str(noT))
36 print("G:"+str(noG))
37 print("C:"+str(noC))
38 file.close()
```

ソースコード 2　1_1.py

実行した結果が以下の通りである．

```
A:59200
T:56195
G:34714
C:34246
```

ソースコード 3　実行結果

1.2　NT_113952.1.fasta の逆相補鎖を生成するプログラムを作成せよ．

以下のように実行する．実行環境は Python2 で，ghostgw 上での動作を確認している．

```
    python 1_2.py /mnt/fs/ohue/newcomer/NT_113952.1.fasta
```

ソースコード 4　実行方法

プログラム本体は以下の通りである．

```
#2019/04/05
#Yuki Tsushima
#tsushima@bi.c.titech.ac.jp
#For Python 2.7

import sys
from rev import rev
from combine import combine

#file open
args = sys.argv
if len(args) != 2:
    sys.stderr.write('Usage: python 1_2.py ./NT_113952.1.fasta\n')
    sys.exit()
path = args[1]
try:
    file = open(path)
except:
    sys.stderr.write('Error in opening file.\n')
    sys.exit()
lines = file.readlines()

#Skip Header
lines.pop(0)
text = combine(lines)
#Reverse
reverse = rev(text)
print(reverse)

file.close()
```

ソースコード 5　1_2.py

```
#2019/04/05
#Yuki Tsushima
#tsushima@bi.c.titech.ac.jp

def rev(text):
    ans = ''
    for char in text:
        if char == 'A':
            ans = 'T' + ans
        elif char == 'T':
            ans = 'A' + ans
        elif char == 'G':
            ans = 'C' + ans
        elif char == 'C':
            ans = 'G' + ans
    return ans
```

ソースコード 6　rev.py

```
#2019/04/05
#Yuki Tsushima
#tsushima@bi.c.titech.ac.jp

def combine(file):
    ans = ''
    for line in file:
        ans = ans + line.strip()
    return ans
```

ソースコード 7　combine.py

実行結果は result_1_2.txt として別添する．

1.3 ウィンドウ幅 w，ステップ幅 s で，ウィンドウ内の GC 含量を出力するプログラムを作成し，NT_113952.1.fasta に $w = 1000$，$s = 300$ で適用した結果を gnuplot 等 Excel 以外のツールでプロットせよ．

以下のように実行する．実行環境は Python2 で，ghostgw 上での動作を確認している．

```
    python 1_3.py /mnt/fs/ohue/newcomer/NT_113952.1.fasta 1000 300 > data.txt
```

ソースコード 8　実行方法

プログラム本体は以下の通りである．また，7 も使用する．

```
#2019/04/11
#Yuki Tsushima
#tsushima@bi.c.titech.ac.jp
#For Python 2.7

import sys
from combine import combine

#file open
args = sys.argv
if len(args) != 4:
    sys.stderr.write('Usage: python 1_3.py ./NT_113952.1.fasta 1000 300\n')
    sys.exit()
path = args[1]
try:
    file = open(path)
except:
    sys.stderr.write('Error in opening file.\n')
    sys.exit()
lines = file.readlines()

#Skip Header
lines.pop(0)
text = combine(lines)
length = len(text)

try:
    w = int(args[2])
    s = int(args[3])
except ValueError as e:
    sys.stderr.write('Incorrect w or s in command line args\n')
    sys.exit()

GC = []
bottom = 0
while bottom < length:
    part = text[bottom:bottom+w]
    if len(part) == 0:
        break
    GC.append((part.count('C') + part.count('G'))/float(len(part)))
    bottom = bottom + s
#Export for gnuplot
for i in range(len(GC)):
    print(str(i*300)+' '+str(GC[i]))

file.close()
```

ソースコード 9　1_3.py

実行結果を gnuplot で描画した結果以下のようになった．

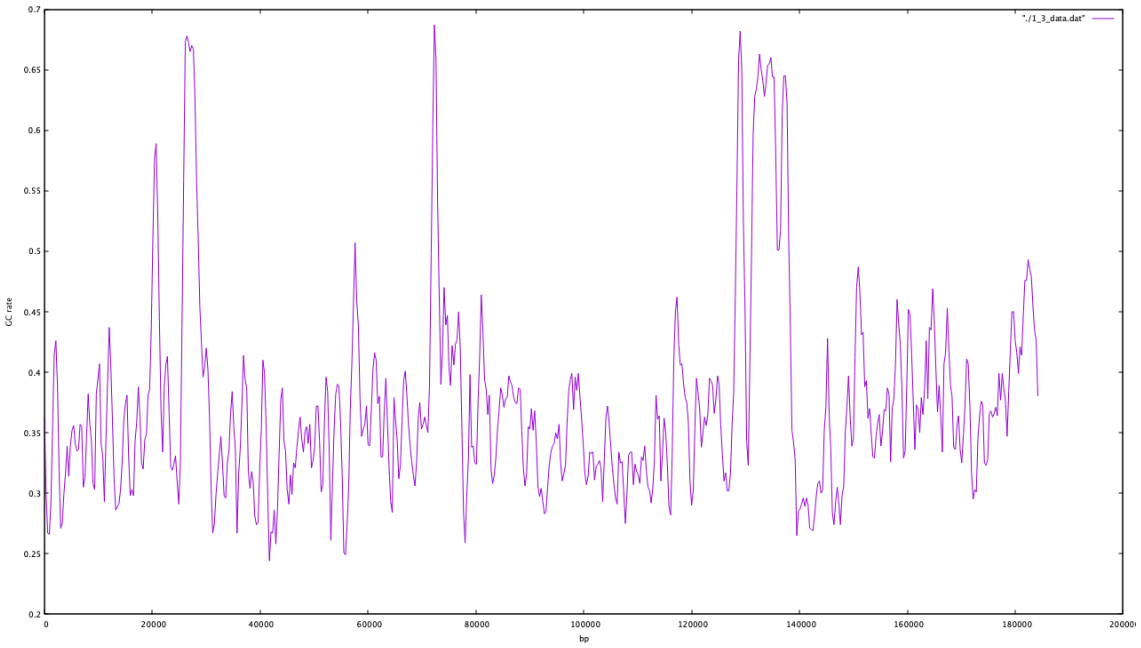

図 1　グラフ

1.4　引数で与えられた部分配列を検索し，何文字目に現れたかを表示するプログラムを作成せよ (逆相補鎖上も検索すること)．部分配列を GAATTC および ATG とし，NT_113952.1.fasta に適用せよ．

以下のように実行する．実行環境は Python2 で，ghostgw 上での動作を確認している．

```
    python 1_4.py /mnt/fs/ohue/newcomer/NT_113952.1.fasta GAATTC
    python 1_4.py /mnt/fs/ohue/newcomer/NT_113952.1.fasta ATG
```

ソースコード 10　実行方法

プログラム本体は以下の通りである．また，6 と 7 も使用する．

```
#2019/04/11
#Yuki Tsushima
#tsushima@bi.c.titech.ac.jp
#For Python 2.7

import sys
from combine import combine
from rev import rev
from findall import findall

#file open
args = sys.argv
if len(args) != 3:
    sys.stderr.write('Usage: python 1_4.py ./NT_113952.1.fasta GAATTC\n')
    sys.exit()
path = args[1]
try:
    file = open(path)
```

```
except:
    sys.stderr.write('Error in opening file.\n')
    sys.exit()
lines = file.readlines()

lines.pop(0)
text = combine(lines)
reverse = rev(text)
index1 = findall(text, args[2])
index2 = findall(reverse, args[2])
print(args[2]+' in STRING is below:')
print(index1)
print(args[2]+' in REVERSE STRING is below:')
print(index2)
file.close()
```

ソースコード 11　1_4.py

```
#2019/04/08
#Yuki Tsushima
#tsushima@bi.c.titech.ac.jp

def findall(text, query):
    index = []
    data = -1
    while True:
        data = text.find(query, data+1)
        if data == -1:
            break
        index.append(data)
    return index
```

ソースコード 12　findall.py

実行した結果が以下の通りである．ただし, ATG の結果については極端に長いため result_1_4_ATG.txt として別添する．

```
GAATTC in STRING is below:
[395, 4319, 5056, 5334, 10567, 14092, 14296, 15029, 19284, 22535, 25293, 25897, 30824, 34194,
  34625, 37333, 56600, 57464, 58202, 60553, 65241, 66725, 79939, 80725, 82012, 86322, 92914,
  94554, 96877, 97628, 99894, 102214, 103133, 120291, 121414, 121612, 124893, 124901, 125389,
  151452, 151874, 156137, 158738, 166310, 175826, 177983, 180004, 182341]
GAATTC in REVERSE STRING is below:
[2008, 4345, 6366, 8523, 18039, 25611, 28212, 32475, 32897, 58960, 59448, 59456, 62737, 62935,
  64058, 81216, 82135, 84455, 86721, 87472, 89795, 91435, 98027, 102337, 103624, 104410,
  117624, 119108, 123796, 126147, 126885, 127749, 147016, 149724, 150155, 153525, 158452,
  159056, 161814, 165065, 169320, 170053, 170257, 173782, 179015, 179293, 180030, 183954]
```

ソースコード 13　実行結果 (GAATTC)

1.5　NT_113952.1.fasta を 6 つの読み枠でアミノ酸配列に変換せよ．Stop コドンはアンダースコアで表示すること．

以下のように実行する．実行環境は Python2 で，ghostgw 上での動作を確認している．

```
    python 1_5.py /mnt/fs/ohue/newcomer/NT_113952.1.fasta
```

ソースコード 14　実行方法

プログラム本体は以下の通りである．また，6 と 7 も使用する．

```
#2019/04/11
#Yuki Tsushima
#tsushima@bi.c.titech.ac.jp
#For Python 2.7

import sys
from combine import combine
from rev import rev
from decode import decode
```

```

#file open
args = sys.argv
if len(args) != 2:
    sys.stderr.write('Usage: python 1_5.py ./NT_113952.1.fasta\n')
    sys.exit()
path = args[1]
try:
    file = open(path)
except:
    sys.stderr.write('Error in opening file.\n')
    sys.exit()
lines = file.readlines()

lines.pop(0)
text = combine(lines)
reverse = rev(text)
print('PEPTIDE 1')
print(decode(text,0))
print('PEPTIDE 2')
print(decode(text,1))
print('PEPTIDE 3')
print(decode(text,2))
print('PEPTIDE 4')
print(decode(reverse,0))
print('PEPTIDE 5')
print(decode(reverse,1))
print('PEPTIDE 6')
print(decode(reverse,2))
file.close()
```

ソースコード 15　1_5.py

```
#2019/04/08
#Yuki Tsushima
#tsushima@bi.c.titech.ac.jp

import sys

def decode(text, offset):
    length = len(text)
    #if too short
    if length < offset + 3:
        return ''
    peptide = ''
    CODON = {'TTT':'Phe', 'TTC':'Phe', 'TTA':'Leu', 'TTG':'Leu',\
             'TCT':'Ser', 'TCC':'Ser', 'TCA':'Ser', 'TCG':'Ser',\
             'TAT':'Tyr', 'TAC':'Tyr', 'TAA':'_'  , 'TAG':'_'  ,\
             'TGT':'Cys', 'TGC':'Cys', 'TGA':'_'  , 'TGG':'Trp',\
             'CTT':'Leu', 'CTC':'Leu', 'CTA':'Leu', 'CTG':'Leu',\
             'CCT':'Pro', 'CCC':'Pro', 'CCA':'Pro', 'CCG':'Pro',\
             'CAT':'His', 'CAC':'His', 'CAA':'Gln', 'CAG':'Gln',\
             'CGT':'Arg', 'CGC':'Arg', 'CGA':'Arg', 'CGG':'Arg',\
             'ATT':'Ile', 'ATC':'Ile', 'ATA':'Ile', 'ATG':'Met',\
             'ACT':'Thr', 'ACC':'Thr', 'ACA':'Thr', 'ACG':'Thr',\
             'AAT':'Asn', 'AAC':'Asn', 'AAA':'Lys', 'AAG':'Lys',\
             'AGT':'Ser', 'AGC':'Ser', 'AGA':'Arg', 'AGG':'Arg',\
             'GTT':'Val', 'GTC':'Val', 'GTA':'Val', 'GTG':'Val',\
             'GCT':'Ala', 'GCC':'Ala', 'GCA':'Ala', 'GCG':'Ala',\
             'GAT':'Asp', 'GAC':'Asp', 'GAA':'Glu', 'GAG':'Glu',\
             'GGT':'Gly', 'GGC':'Gly', 'GGA':'Gly', 'GGG':'Gly'}
    start = offset
    end = start + 3
    try:
        #Skip to Met
        while end <= length:
            if CODON[text[start:end]] == 'Met':
                break
            start = end
            end = start + 3
        #Decode
        while end <= length:
            peptide = peptide + ' ' + CODON[text[start:end]]
            if peptide[len(peptide)-1] == '_':
                break
            start = end
            end = start + 3
        return peptide.strip()
    except (KeyError):
        sys.stderr.write('Unknown CODON Error\n')
```

```
48          return ''
```

ソースコード 16　decode.py

実行した結果が以下の通りである．

```
1  PEPTIDE 1
2  Met Ile Val Met Asn Ser Asn Cys Cys Leu Cys Arg Pro Thr Arg Phe Leu Thr Ser Leu Ser Tyr His Phe
       Leu Leu Ser Tyr Leu Leu Ser Lys Cys Ile Gln Met Lys Gly Cys Gly Glu Cys _
3  PEPTIDE 2
4  Met Pro Arg Glu Ile Ser Arg Ser Ser Val Pro Cys _
5  PEPTIDE 3
6  Met Pro _
7  PEPTIDE 4
8  Met Leu Arg Thr Leu Leu Leu Ile Val _
9  PEPTIDE 5
10 Met Gln Asn Lys Phe Ile Arg His _
11 PEPTIDE 6
12 Met Gly Arg Lys Asp Lys Ala Ala Ile _
```

ソースコード 17　実行結果

2　タンパク質構造情報処理

2.1　PyMOL ソフトウェアを利用し，ヒトのヘモグロビンの構造をチェインごとに異なる色で表示し，4 量体であることを確認せよ．また，A チェインだけを表示し，タンパク質鎖を cartoon 表示して 2 次構造に従って色付けし，結合するヘムを stick 表示して原子ごとに色分けせよ．

チェインごとに色分けしたヘモグロビンが画像 2 である．

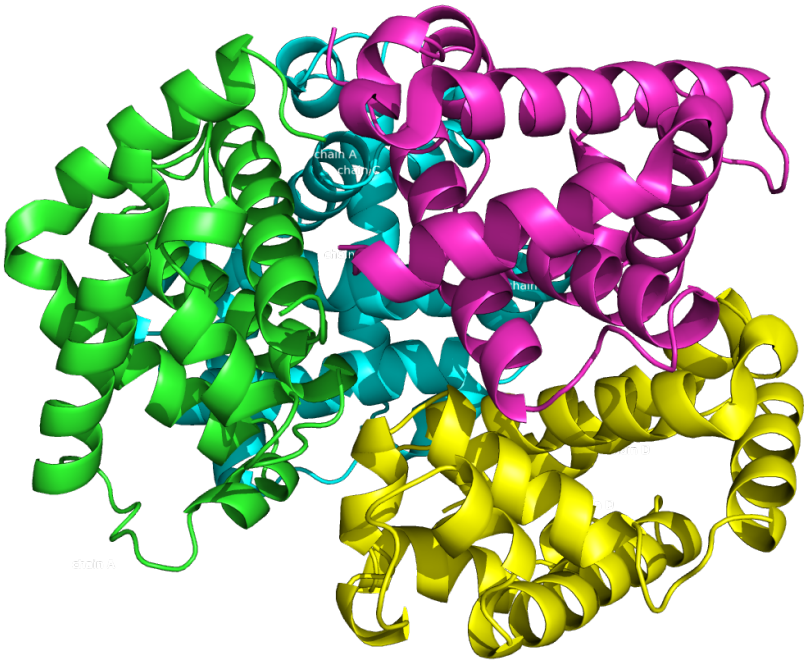

図 2　ヘモグロビン

また，A チェインとヘムを表示したものが画像 3 である．

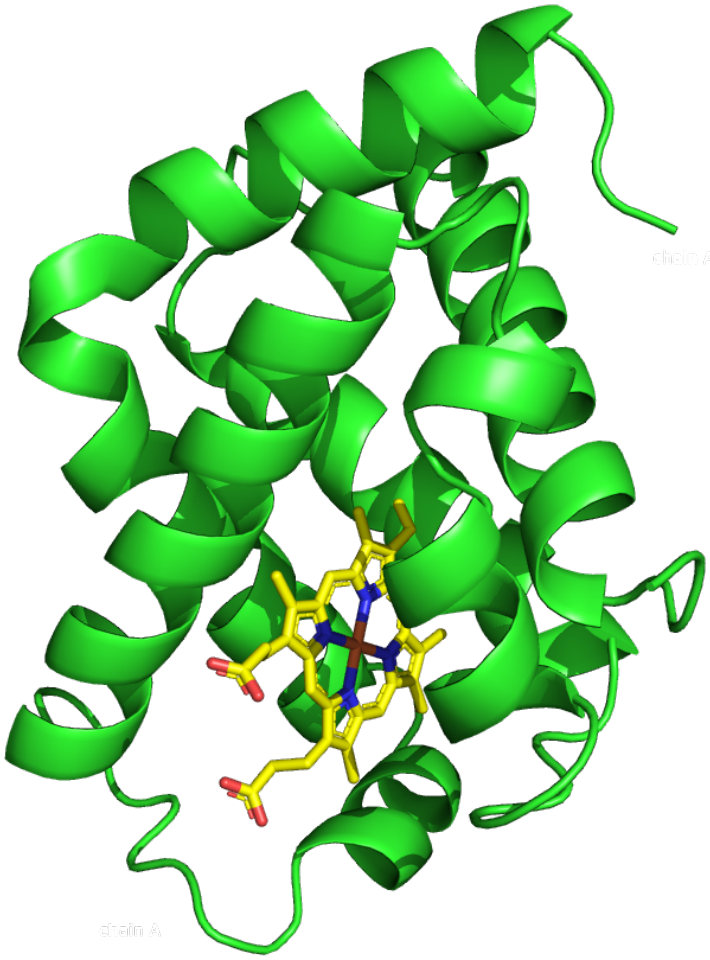

図 3　A チェインとヘム

2.2　PDB ファイル名とチェイン名を引数にして，その回転半径を計算するプログラムを作成せよ．

2.3　上記のプログラムの結果を利用し，PyMOL で重心から回転半径の範囲内にある原子を赤で，範囲外にある原子を青で色付けせよ．

この 2 つは一緒に行い，以下のプログラムを作成した．実行環境は Anaconda3 の Python 2.7 環境である．

```
#2019/04/17
#Yuki Tsushima
#tsushima@bi.c.titech.ac.jp
#For Python 2.7

import math
import sys
import pymol
```

```
9
10 #file open
11 args = sys.argv
12 if len(args) != 3:
13     sys.stderr.write('Usage: python 2_7.py ./1BUW.pdb A\n')
14     sys.exit()
15 path = args[1]
16 try:
17     file = open(path)
18 except:
19     sys.stderr.write('Error in opening file.\n')
20     sys.exit()
21 lines = file.readlines()
22
23 WEIGHT =\
24 { 'H':1.008,\
25  'LI':6.941, 'BE':9.012,  'B':10.81,  'C':12.01,  'N':14.01,  'O':16.00,  'F':19.00,\
26  'NA':22.99, 'MG':24.31, 'AL':26.98, 'SI':28.09,  'P':30.97,  'S':32.07, 'CL':35.45,}
27 total_weight = 0
28 x_weight = 0
29 y_weight = 0
30 z_weight = 0
31 atom_data = []
32 for line in lines:
33     if line[0:6] == 'ATOM  ' and line[21] == args[2]:
34         #Parse Data
35         atom_number = int(line[6:11])
36         atom_name = line[12:16].strip()
37         #identifier = line[16].strip()
38         #res_name = line[17:20].strip()
39         chain = line[21].strip()
40         res_number = int(line[22:26])
41         #res_code = line[26].strip()
42         x = float(line[30:38])
43         y = float(line[38:46])
44         z = float(line[46:54])
45         #occupy = float(line[54:60])
46         #tmp = line[60:66].strip()
47         elem_symbol = line[76:78].strip()
48         #charge = line[78:80].strip()
49
50         #Calculate Center of Grabity
51         total_weight = total_weight + WEIGHT[elem_symbol]
52         x_weight = x_weight + x * WEIGHT[elem_symbol]
53         y_weight = y_weight + y * WEIGHT[elem_symbol]
54         z_weight = z_weight + z * WEIGHT[elem_symbol]
55
56         #Register Atom Data
57         atom_data.append([x, y, z, atom_number, atom_name, elem_symbol, chain, res_number])
58 #IF no Data
59 if total_weight == 0:
60     sys.stderr.write('Chain '+args[2]+' does not exist.\n')
61     sys.exit()
62
63 #Calculate Center of Grabity
64 x_balance = x_weight / total_weight
65 y_balance = y_weight / total_weight
66 z_balance = z_weight / total_weight
67
68 #Calculate circle r
69 r = 0
70 for data in atom_data:
71     r = r + math.sqrt((data[0]-x_balance)**2+(data[1]-y_balance)**2+(data[2]-z_balance)**2)
72 r = r / len(atom_data)
73 print('Center: ('+str(x_balance)+', '+str(y_balance)+', '+str(z_balance)+')')
74 print('Radius: '+str(r))
75
76 #identify color
77 red = []
78 blue = []
79 for data in atom_data:
80     if math.sqrt((data[0]-x_balance)**2+(data[1]-y_balance)**2+(data[2]-z_balance)**2) > r:
81         blue.append(data)
82     else:
83         red.append(data)
84
85 #Launch PyMOL
86 pymol.finish_launching()
87 pymol.cmd.load(path)
88 pymol.cmd.do("hide everything")
89 pymol.cmd.do("show stick, chain "+args[2])
```

```
90 #set color
91 for data in red:
92     pymol.cmd.do("color red, (name "+data[4]+" and resi "+str(data[7])+" and chain "+data
       [6]+")")
93 for data in blue:
94     pymol.cmd.do("color blue, (name "+data[4]+" and resi "+str(data[7])+" and chain "+data
       [6]+")")
95 file.close()
```

ソースコード 18　2_7.py

このプログラムを次のように実行する．

```
1     python 2_7.py ./1BUW.pdb A
```

ソースコード 19　実行方法

結果は次の通りである．

```
1 Center: (47.0059686237, 33.4827138428, 35.7021762506)
2 Radius: 14.0781227416
```

ソースコード 20　実行結果

また，描画結果として以下の画像を得た．

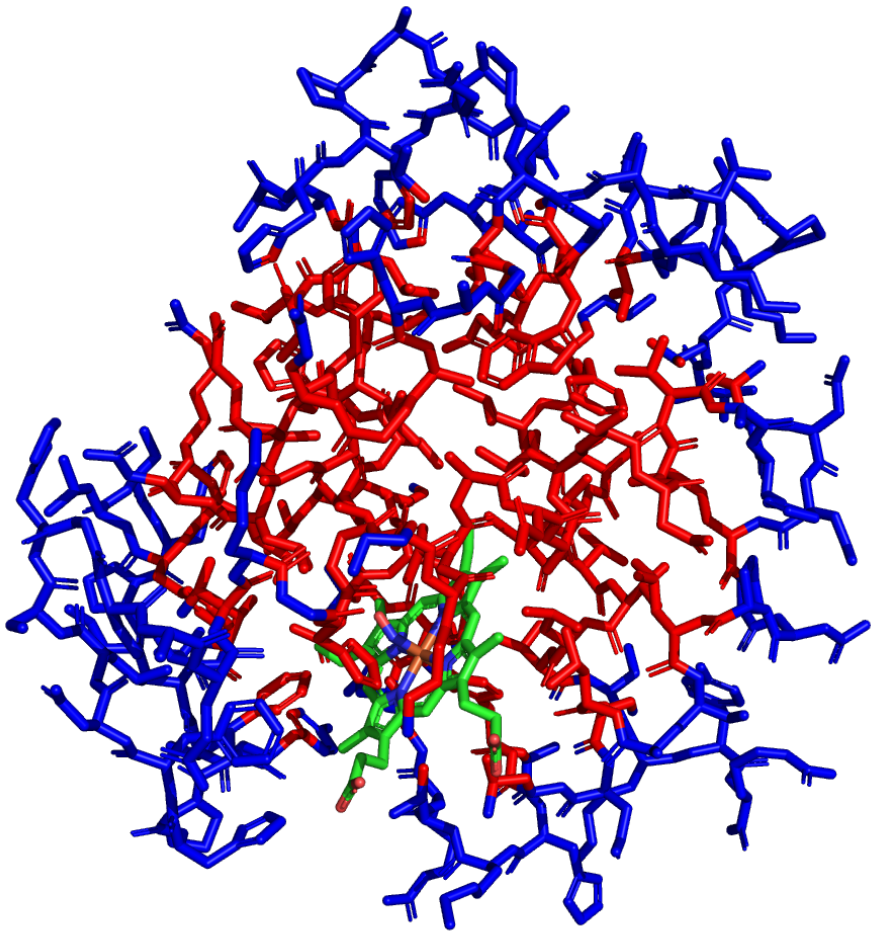

図 4　回転半径

3 機械学習

3.1 Python の scikit-learn の SVM で ionosphere のデータの 10-fold Cross Validation を実施せよ．予測性能として precision, recall, MCC, ROC 曲線の AUC 値 (AUROC), F-score を求めよ．カーネルは RBF カーネルとし，パラメータは適宜定めよ．

作成したプログラムを 21 に示す．Anaconda3 の Python2.7 環境で動作する．

```
#2019/04/16
#Yuki Tsushima
#tsushima@bi.c.titech.ac.jp
#For Python 2.7

import sys
from sklearn.svm import SVC
from sklearn.metrics import precision_score, recall_score, f1_score, matthews_corrcoef,
  roc_auc_score
from load_ionosphere import load_ionosphere
from sklearn.model_selection import *

#TP, TN : True
#FN : actually positive, but expected negative
#FP : actually negative, but expected positive
#Precision : TP / TP + FP
    # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html#
      sklearn.metrics.precision_score
#Recall    : TP / TP + FN
    # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html#
      sklearn.metrics.recall_score
#F-measure : 2 * Recall * Precision / Recall + Precision
    # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html#sklearn.
      metrics.f1_score
#MCC : Matthews Correlation Coefficient
    # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews_corrcoef.html#
      sklearn.metrics.matthews_corrcoef
#AUROC :
    # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html#
      sklearn.metrics.roc_auc_score

svc = SVC(C=1.0, kernel='rbf', gamma=0.2)
kf = KFold(n_splits=10, shuffle=True)
[target, parameters] = load_ionosphere()
counter = 0
print('Set\tPreci.\tRecall\tMCC\tAUROC\tF-score')
for train_index, test_index in kf.split(parameters, target):
    counter = counter + 1
    train_param  = []
    train_target = []
    test_param   = []
    test_target  = []
    for i in range(0,len(train_index)):
        train_param.append(parameters[train_index[i]])
        train_target.append(target[train_index[i]])
    for i in range(0,len(test_index)):
        test_param.append(parameters[test_index[i]])
        test_target.append(target[test_index[i]])
    sys.stdout.write(str(counter)+'\t')
    learn = svc.fit(train_param, train_target)
    print(str(precision_score(test_target, learn.predict(test_param)))[0:7]+'\t'\
         +str(recall_score(test_target, learn.predict(test_param)))[0:7]+'\t'\
         +str(matthews_corrcoef(test_target, learn.predict(test_param)))[0:7]+'\t'\
         +str(roc_auc_score(test_target, learn.predict(test_param)))[0:7]+'\t'\
         +str(f1_score(test_target, learn.predict(test_param)))[0:7])
```

ソースコード 21　3_9.py

また，ionosphere.scale の読み込みにあたって，欠番のパラメータを 0 で補完するなどの処理を行うため，読み込み部分は別のプログラム 22 として分離した．

```
#2019/04/16
#Yuki Tsushima
#tsushima@bi.c.titech.ac.jp
```

```
#For Python 2.7

import sys

def load_ionosphere():
    #file open
    try:
        file = open('./ionosphere.scale')
    except:
        sys.stderr.write('Error in opening file.\n')
        sys.exit()

    lines = file.readlines()
    target = []
    parameters = []
    for line in lines:
        set = line.split()
        #target is 1 or 0
        target.append(int(set.pop(0))+1//2)
        datas = {}
        params = []
        for data in set:
            datas[int(data.split(':')[0])] = float(data.split(':')[1])
        for i in range(0,34):
            try:
                params.append(datas[i+1])
            except KeyError:
                params.append(0)
        parameters.append(params)
    file.close()
    return [target, parameters]
```

ソースコード 22　load_ionosphere.py

実行した結果を 23 に示す．

```
Set     Preci.  Recall  MCC     AUROC   F-score
1       1.0     1.0     1.0     1.0     1.0
2       1.0     0.95454 0.94146 0.97727 0.97674
3       0.9     1.0     0.89113 0.94117 0.94736
4       0.84    0.95454 0.69186 0.82342 0.89361
5       0.92    1.0     0.87559 0.91666 0.95833
6       1.0     1.0     1.0     1.0     1.0
7       0.91666 0.95652 0.80760 0.89492 0.93617
8       1.0     0.9375  0.75    0.96875 0.96774
9       1.0     0.95238 0.94280 0.97619 0.97560
10      0.90476 0.95    0.82495 0.90833 0.92682
```

ソースコード 23　実行結果

3.2　ionosphere データにおいて，より良い RBF カーネルパラメータ γ とコストパラメータ C の値を探索せよ．評価方法は 10-fold Cross Validation とし，評価基準は AUROC と F-score の 2 通りを試すこと．

作成したプログラムを 24 に示す．Anaconda3 の Python2.7 環境で動作する．

```
#2019/04/16
#Yuki Tsushima
#tsushima@bi.c.titech.ac.jp
#For Python 2.7

import sys
from sklearn.svm import SVC
from sklearn.metrics import precision_score, recall_score, f1_score, matthews_corrcoef,
  roc_auc_score
from load_ionosphere import load_ionosphere
from sklearn.model_selection import *

#Help https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.
  html
#Help https://scikit-learn.org/stable/modules/model_evaluation.html#scoring-parameter

[target, parameters] = load_ionosphere()
```

```
param_grid = {'C':[0.001, 0.01, 0.1, 1, 10, 100, 1000], 'gamma':[0.001, 0.01, 0.1, 1, 10, 100, 1000]}
print('Evaluate with AUROC')
search_auroc = GridSearchCV(estimator=SVC(), param_grid=param_grid, scoring='roc_auc',cv=10, iid=False).fit(parameters, target)
print(search_auroc.best_params_)
print(search_auroc.best_score_)
print('Evaluate with F-score')
search_f = GridSearchCV(estimator=SVC(), param_grid=param_grid, scoring='f1',cv=10, iid=False).fit(parameters, target)
print(search_f.best_params_)
print(search_f.best_score_)
```

ソースコード 24　3_10.py

実行した結果を 25 に示す.

```
Evaluate with AUROC
{'C': 10, 'gamma': 1}
0.9860481909394953
Evaluate with F-score
{'C': 10, 'gamma': 0.1}
0.9616653503831382
```

ソースコード 25　実行結果

4　創薬情報処理・機械学習

4.1　(準備 1)Python に RDKit をインストールし，化合物ファイル (SDF ファイル) を読み込んで構造式が出力できることを確認せよ.

省略する.

4.2　(準備 2) 論文 Leung SSF, et al. J Chem Inf Model 56: 924-020, 2016. の Supporting Information より，3D SDF ファイル (TXT) と PDF ファイルをダウンロードせよ．Table S1 の実験値「RRCK Log P_{app}」の値をパース (転記) し，CSV ファイル等で準備せよ.

省略する.

4.3　(準備 3) 所望の化合物に対し，RDKit の ECFP4 fingerprint を計算できるようにせよ.

calc_fgprint 関数を作成した。コードは 26 の通りである。

```
#2019/04/15
#Yuki Tsushima
#tsushima@bi.c.titech.ac.jp
#For Python 3.7

#User rdkit-env

from rdkit import Chem
from rdkit.Chem import AllChem

def calc_fgprint(mol):
    data = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits = 1024)
    set = []
    for i in range(0,1024):
        if data.GetBit(i):
            set.append(1)
        else:
```

```
            set.append(0)
    return set
```

ソースコード 26　calc_fgprint.py

4.4　RRCK Log P_{app} を目的関数，ECFP4 fingerprint を特徴ベクトルとして，Data Set 3(医薬品 104 化合物) に関して回帰 (10-fold Cross Validation) を行う機械学習プログラムを作成せよ．学習器は Support Vector Regression とし，カーネルは RBF カーネルとすること．

スクリプト 27 を作成した。

```
#2019/04/15
#Yuki Tsushima
#tsushima@bi.c.titech.ac.jp
#For Python 3.7

#User rdkit-env

import csv
import sys
from calc_fgprint import calc_fgprint
from rdkit import rdBase, Chem
from rdkit.Chem import AllChem, Draw
from rdkit.Chem.Draw import rdMolDraw2D

from sklearn import *
from sklearn.svm import SVR
from sklearn.model_selection import *

#Load Mols
suppl = Chem.SDMolSupplier('./ci6b00005_si_002.txt', removeHs=False) #Supplier
mols = [x for x in suppl if x is not None]
mols_set3 = mols[106:210]
fingerprint = []
for data in mols_set3:
    fingerprint.append(calc_fgprint(data))

#Load RRCKs
csv_file = open('./dataset3.csv', 'r')
f = csv.reader(csv_file)
RRCK_set3 = []
for row in f:
    RRCK_set3.append(float(row[0]))

svr = SVR(kernel='rbf', C=100, gamma=0.01)

score = cross_val_score(estimator=svr, X=fingerprint, y=RRCK_set3, cv=10)
print(sum(score)/len(score))
```

ソースコード 27　4_14.py

実行結果は 28 の通りである。

```
0.2630136419201868
```

ソースコード 28　実行結果

4.5　10-fold Cross Validation によって，RRCK Log P_{app} の予測値との平均 2 乗誤差 (RMSE) が最も小さくなるパラメータを探索せよ．またそのときの RMSE と R^2 値を求めよ．

スクリプト 29 を作成した。

```
#2019/04/18
```

```
#Yuki Tsushima
#tsushima@bi.c.titech.ac.jp
#For Python 3.7

#User rdkit-env

import csv
import sys
from calc_fgprint import calc_fgprint
from rdkit import rdBase, Chem

from sklearn import svm, model_selection, metrics
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

#Load Mols
suppl = Chem.SDMolSupplier('./ci6b00005_si_002.txt', removeHs=False) #Supplier
mols = [x for x in suppl if x is not None]
mols_set3 = mols[106:210]
fingerprint = []
for data in mols_set3:
    fingerprint.append(calc_fgprint(data))

#Load RRCKs
csv_file = open('./dataset3.csv', 'r')
f = csv.reader(csv_file)
RRCK_set3 = []
for row in f:
    RRCK_set3.append(float(row[0]))

param_grid = {'C':[0.001, 0.01, 0.1, 1, 10, 100, 1000], 'gamma':[0.001, 0.01, 0.1, 1, 10, 100, 1000]}
def rmse(y, y_pred):
    return np.sqrt(mean_squared_error(y, y_pred))
search = GridSearchCV(estimator=SVR(), param_grid=param_grid, scoring='neg_mean_squared_error', cv=10, iid=False).fit(fingerprint, RRCK_set3)
print('R2_score : {}'.format(r2_score(RRCK_set3, search.predict(fingerprint))))
print('RMSE : {}'.format(rmse(RRCK_set3, search.predict(fingerprint))))
print('best_params : {}'.format(search.best_params_))
```

ソースコード 29　4_15.py

実行結果は 30 の通りである。

```
R2_score : 0.9747505807591622
RMSE : 0.10158867737298335
best_params : {'C': 100, 'gamma': 0.01}
```

ソースコード 30　実行結果

4.6 (15) で決めたパラメータの学習器で，Data Set 1(環状ペプチド 7 化合物)，Data Set 4(環状ペプチド 16 化合物)，Data Set 8(環状ペプチド 22 化合物) の RRCK Log P_{app} の予測値を求めよ．それぞれの Data Set ごとに RMSE と R^2 値を求め，予測値と実験値の散布図を描け．

スクリプト 31 を作成した。

```
#2019/04/18
#Yuki Tsushima
#tsushima@bi.c.titech.ac.jp
#For Python 3.7

#User rdkit-env

import codecs
import csv
import sys
from calc_fgprint import calc_fgprint
from rdkit import rdBase, Chem

```

```
from sklearn import svm, model_selection, metrics
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

#Load Mols
suppl1 = Chem.SDMolSupplier('./dataset1.sdf', removeHs=False) #Supplier
suppl3 = Chem.SDMolSupplier('./ci6b00005_si_002.txt', removeHs=False) #Supplier
suppl4 = Chem.SDMolSupplier('./dataset4.sdf', removeHs=False) #Supplier
suppl8 = Chem.SDMolSupplier('./dataset8.sdf', removeHs=False) #Supplier
mols = [x for x in suppl3 if x is not None]
mols_set3 = mols[106:210]
mols_set1 = [x for x in suppl1 if x is not None]
mols_set4 = [x for x in suppl4 if x is not None]
mols_set8 = [x for x in suppl8 if x is not None]
finger1 = []
finger3 = []
finger4 = []
finger8 = []
for data in mols_set1:
    finger1.append(calc_fgprint(data))
for data in mols_set3:
    finger3.append(calc_fgprint(data))
for data in mols_set4:
    finger4.append(calc_fgprint(data))
for data in mols_set8:
    finger8.append(calc_fgprint(data))

#Load RRCKs
csv1 = open('./dataset1.csv', 'r')
f1 = csv.reader(csv1)
RRCK_set1 = []
for row in f1:
    RRCK_set1.append(float(row[0]))
csv3 = open('./dataset3.csv', 'r')
f3 = csv.reader(csv3)
RRCK_set3 = []
for row in f3:
    RRCK_set3.append(float(row[0]))
csv4 = open('./dataset4.csv', 'r')
f4 = csv.reader(csv4)
RRCK_set4 = []
for row in f4:
    RRCK_set4.append(float(row[0]))
csv8 = open('./dataset8.csv', 'r')
f8 = csv.reader(csv8)
RRCK_set8 = []
for row in f8:
    RRCK_set8.append(float(row[0]))

#Learn
svr = SVR(kernel='rbf', C=100, gamma=0.01).fit(finger3, RRCK_set3)
def rmse(y, y_pred):
    return np.sqrt(mean_squared_error(y, y_pred))

#Predict and output
pre1 = svr.predict(finger1)
with codecs.open('predict1.csv', 'w', 'utf-8') as f:
    for i in range(0,len(pre1)):
        f.write(str(RRCK_set1[i])+','+str(pre1[i])+'\n')
print("Dataset 1")
print('R2_score : {}'.format(r2_score(RRCK_set1, pre1)))
print('RMSE : {}'.format(rmse(RRCK_set1, pre1)))
pre3 = svr.predict(finger3)
with codecs.open('predict3.csv', 'w', 'utf-8') as f:
    for i in range(0,len(pre3)):
        f.write(str(RRCK_set3[i])+','+str(pre3[i])+'\n')
print("Dataset 3")
print('R2_score : {}'.format(r2_score(RRCK_set3, pre3)))
print('RMSE : {}'.format(rmse(RRCK_set3, pre3)))
pre4 = svr.predict(finger4)
with codecs.open('predict4.csv', 'w', 'utf-8') as f:
    for i in range(0,len(pre4)):
        f.write(str(RRCK_set4[i])+','+str(pre4[i])+'\n')
print("Dataset 4")
print('R2_score : {}'.format(r2_score(RRCK_set4, pre4)))
print('RMSE : {}'.format(rmse(RRCK_set4, pre4)))
pre8 = svr.predict(finger8)
with codecs.open('predict8.csv', 'w', 'utf-8') as f:
    for i in range(0,len(pre8)):
```

```
        f.write(str(RRCK_set8[i])+','+str(pre8[i])+'\n')
print("Dataset 8")
print('R2_score : {}'.format(r2_score(RRCK_set8, pre8)))
print('RMSE : {}'.format(rmse(RRCK_set8, pre8)))
```

ソースコード 31　4_16.py

実行結果は 32 の通りである。

```
Dataset 1
R2_score : -2.314801681170682
RMSE : 0.6643257009105027
Dataset 3
R2_score : 0.9747505807591622
RMSE : 0.10158867737298335
Dataset 4
R2_score : -2.1150984682038985
RMSE : 0.9198261864774264
Dataset 8
R2_score : 0.12606435549538175
RMSE : 0.4296164950959734
```

ソースコード 32　実行結果

また、実験値と予測値の散布図は出力したファイルから Gnuplot で描画した。図 5、6、7、8 を得た。

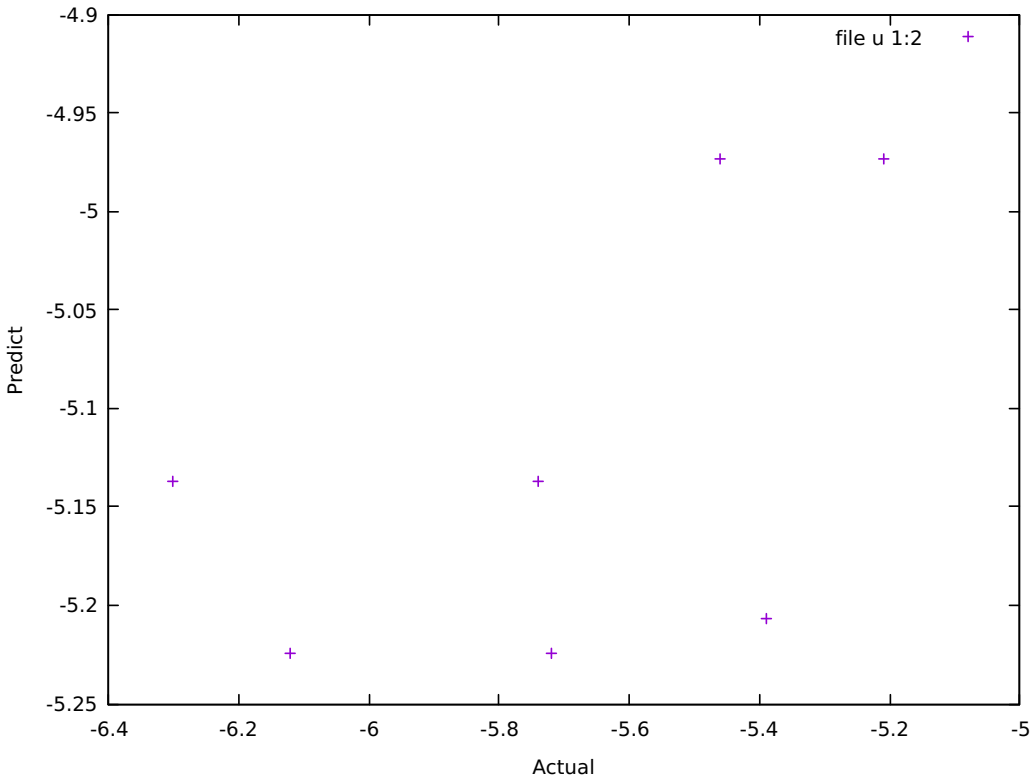

図 5　Data Set 1

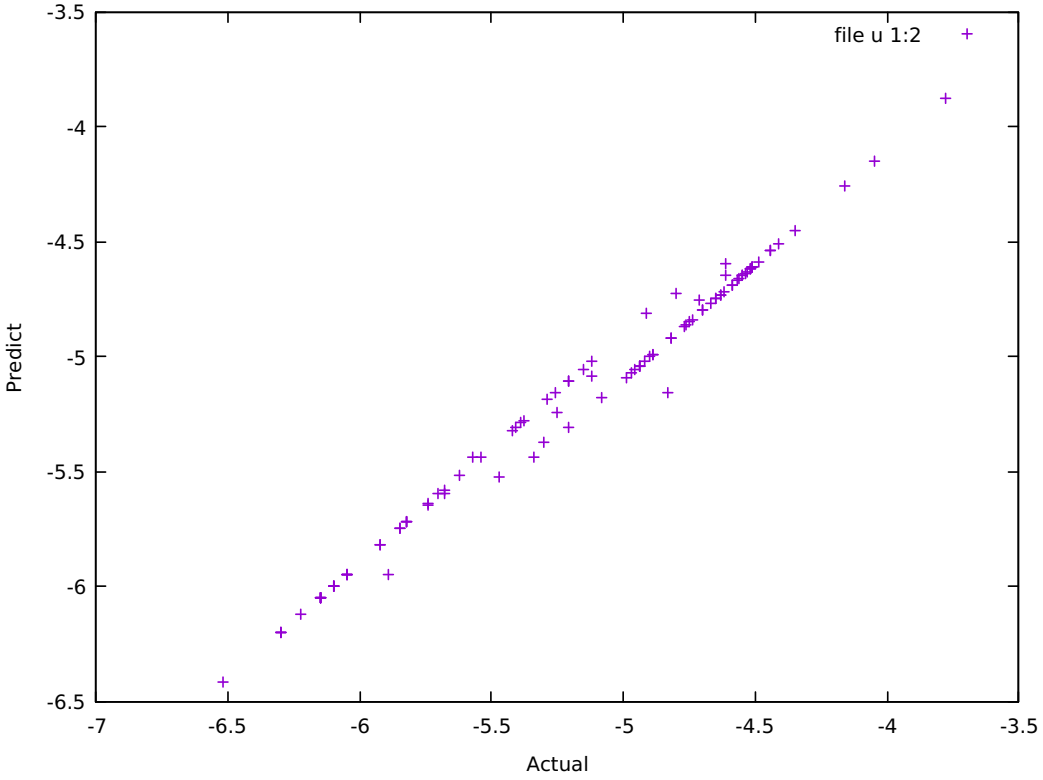

図 6　Data Set 3

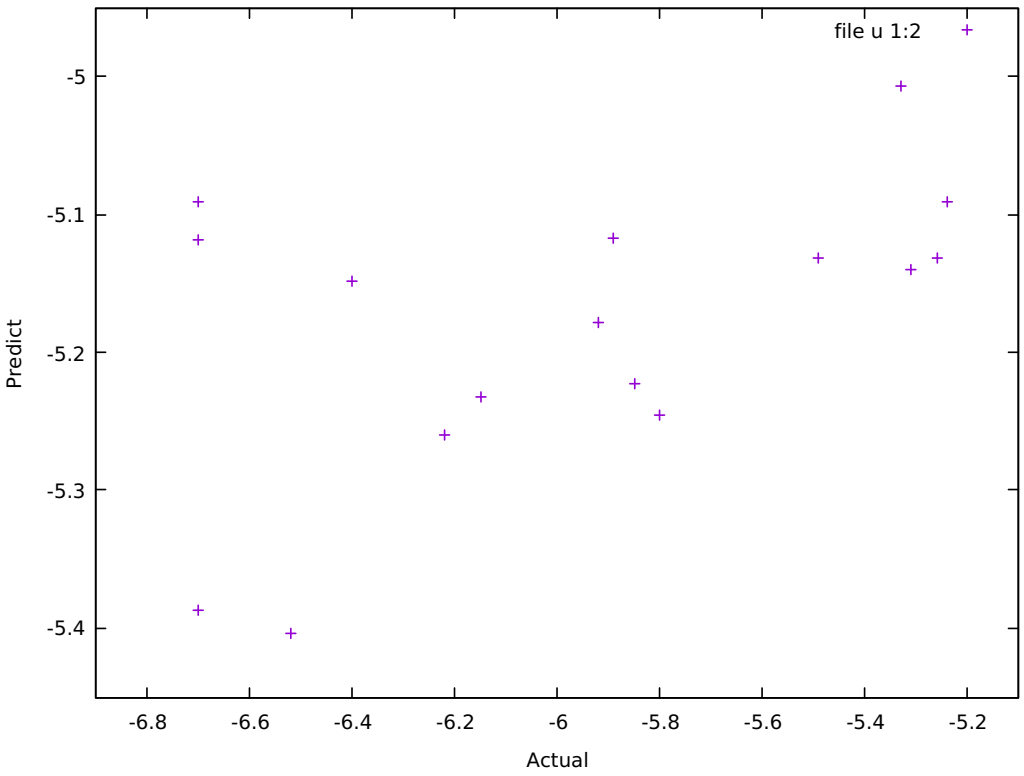

図 7　Data Set 4

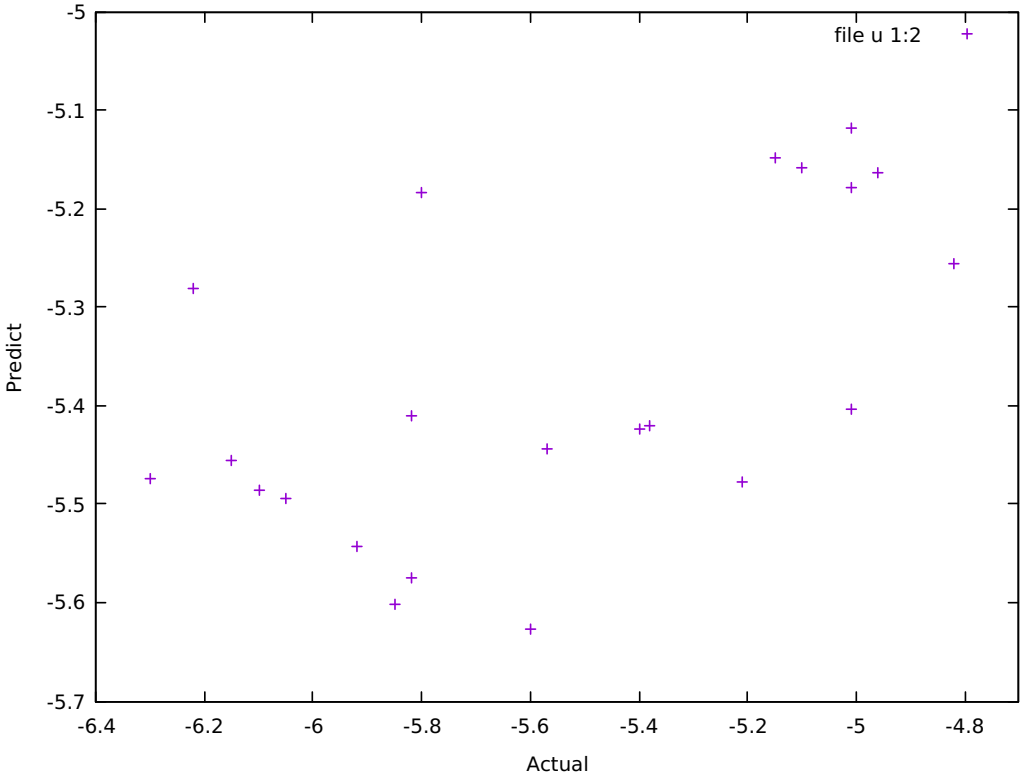
file u 1:2
Predict
Actual
-5
-5.1
-5.2
-5.3
-5.4
-5.5
-5.6
-5.7
-6.4
-6.2
-6
-5.8
-5.6
-5.4
-5.2
-5
-4.8

図 8　Data Set 8