BIOINFORMATIKA:
PREDIKSI STRUKTUR SEKUNDER PROTEIN MENGGUNAKAN SUPPORT VECTOR MACHINE PADA DATASET RS126

VINCENT MICHAEL SUTANTO
16/398531/PA/17492

PROGRAM STUDI ILMU KOMPUTER
DEPARTEMEN ILMU KOMPUTER DAN ELEKTRONIKA
UNIVERSITAS GADJAH MADA
YOGYAKARTA
2019

Link Github:

https://github.com/vincentmichael089/SVM-PSSP

Membaca Dataset

Dataset RS126 dibuka dan diolah dengan menggunakan bahasa pemrograman python. Line ganjil dari data merujuk pada struktur primer protein dan line genap merujuk pada struktur sekunder dari struktur primer di line sebelumya.

```
raw_primer = []
raw_sekunder = []


with open('RS126.data.txt', 'r') as f:
    for count, line in enumerate(f, start=1):
        if count % 2 == 0:
            raw_sekunder.append(line.strip())
        else:
            raw_primer.append(line.strip())
```

Contoh data struktur primer terbaca sebagai berikut:

```
['APAFSVSPASGASDGQSVSVSVAAAGETYYIAQCAPVGGQDACNPATATSFTTDASGAASFSFTVRKSYAGQTPS
GTPVGSVDCATDACNLGAGNSGLNLGHVALTFG',
'CSVDIQGNDQMQFNTNAITVDKSCKQFTVNLSHPGNLPKNVMGHNWVLSTAADMQGVVTDGMASGLDKDYL
KPDDSRVIAHTKLIGSGEKDSVTFDVSKLKEGEQYMFFCTFPGHSALMKGTLTLK',
'NVYHDGACPEVKPVDNFDWSNYHGKWWEVAKYPNSVEKYGKCGWAEYTPEGKSVKVSNYHVIHGKEYFIEGTA
YPVGDSKIGKIYHKLTYGGVTKENVFNVLSTDNKNYIIGYYCKYDEDKKGHQDFVWVLSRSKVLTGEAKTAVENYLIG
SPVVDSQKLVYSDFSEAACKVN',
```

Contoh data struktur sekunder terbaca sebagai berikut:

```
['CCEEEEECCCCCCCCCEEEEEEECCCCEEEEEEECEECCEECCCCCCCCEEECCCCCCCEEEEECCCEEEECCCCCEE
EEEECCCCCCEEEEECCCCCCCCCCCCC',
'CCEEEECCCCCCCCCCCEECCCCCCEEEEEEECCCCCCCCCCCCECCEEEECCCCHHHHHHHHHHCHHHHCCCCCCC
CCCECCCCECCCCCEEEEEECCCCCCCCCEEEECCCCCCCCCCEEEEECC',
'CEEEECCCCCCCCCCCCCHHHCCEEEEEEEECCCCCCCCCEEEEEEEEECCCCEEEEEEEEECCEEEEEEEEEECCCC
CCCEEEEEEEECCEEEEEEEEEEECCCCEEEEEEEEEECCCCEEEEEEEEECCCCCCHHHHHHHHHHHHCCCCCH
HHCEECCCCHHHHCCC',
```

Mencari data yang tidak lengkap:

Dilakukan pengecekan terhadap data apakah terdapat perbedaan jumlah karakter struktur primer dan struktur sekunder (karena prediksi struktur sekunder protein termasuk ke dalam permasalahan *sequence labelling*, maka dipastikan jumlah karakter dari struktur primer dan struktur sekunder selalu sama).

```
for i in range(len(raw_sekunder)):
    len1 = len(raw_sekunder[i])
    len2 = len(raw_primer[i])

    if(len1 != len2):
        print(i," ", raw_sekunder[i]," ",raw_primer[i])
```

Ditemukan bahwa data pada urutan ke 109 memiliki panjang struktur primer dan struktur sekunder yang berbeda. Karena hanya 1 data maka data struktur primer dan struktur sekunder dari urutan ke 109 tersebut dihapus.

Orthogonal Encoding - Target Labeling

Setiap data struktur primer dan struktur sekunder di split sehingga dapat diencode kedalam bentuk orthogonal.

```
def split(sequence):
    return [char for char in sequence]

split_primer = []
split_sekunder = []
for i in range(len(raw_primer)):
    split_primer.append(split(raw_primer[i]))
    split_sekunder.append(split(raw_sekunder[i]))
```

Hasil split struktur primer dan struktur sekunder protein kemudian diubah kedalam bentuk Orthogonal Encoding dan Target Labeling. Cuplikan switch case untuk setiap asam amino pada struktur primer protein sebagai berikut:

```
def orthogonal_primer(arg):
    switch = {
        'A' : np.array([1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]),
        'C' : np.array([0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]),
        'E' : np.array([0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]),
        'D' : np.array([0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]),
        'G' : np.array([0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]),
        'F' : np.array([0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0]),
        'I' : np.array([0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0]),
        'H' : np.array([0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0]),
        'K' : np.array([0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0]),
        'M' : np.array([0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0]),
        'L' : np.array([0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0]),
        'N' : np.array([0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0]),
        'Q' : np.array([0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0]),
        'P' : np.array([0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0]),
        'S' : np.array([0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0]),
        'R' : np.array([0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0]),
        'T' : np.array([0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0]),
        'W' : np.array([0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0]),
        'V' : np.array([0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0]),
        'Y' : np.array([0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1])
    }
    return switch.get(arg)
```

Dan untuk target labeling dibentuk juga fungsi switch case sebagai berikut:

```
def orthogonal_sekunder(arg):
    switch = {
        'H' : 0,
        'C' : 1,
        'E' : 2
    }
    return switch.get(arg)
```

Dataset - Label dan Fitur

Setiap hasil split dari struktur sekunder protein akan dijadikan label / target class.

```
def target(lis):
    Y = []
    for i in range(len(lis)):
        for j    in range(len(lis[i])):
            Y.append(lis[i][j])
     return Y


y_label = target(split_sekunder)
```

Dihasilkan total keseluruhan label berjumlah 22.594. Angka ini akan digunakan untuk memastikan bahwa dataset juga memiliki fitur sebanyak 22.594 data juga.

```
def window_padding_data(size, sequence):
    num = int(size/2)
    zeros = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
    for i in range(len(sequence)):
        for j in range(num):
            sequence[i].append(zeros)
            sequence[i].insert(0, zeros)


    X = []
    temp = []

    for k in range(len(sequence)):
        for l in range(len(sequence[k])-(size-1)):
            temp = sequence[k][l:l+5]
            X.append(temp)
            temp = []
    return X
```

Pembentukan fitur data dilakukan dengan menggunakan fungsi window_padding_data. Fungsi ini akan menerima ukuran dari sliding window dan sekuens struktur primer protein. Di fungsi ini fitur akan diolah seperti penambahan padding 0 di awal dan diakhir dan pengambilan fitur-fitur hasil windowing sehingga data keluaran bisa langsung dilatih pada model SVM.

Contoh penggunaan fungsi window_padding_data adalah sebagai berikut:

```
X = window_padding_data(5,split_primer)
```

Dimana data struktur primer protein akan diolah dengan ukuran sliding window SVM sebesar 5. Sebelum dimasukkan ke model SVM Scikit-Learn, data di reshape untuk mengikuti ukuran input dari model.

```
X = np.array(X)
y_label = np.array(y_label)
X = X.reshape(22594, 5*20)
```

Data direshape menjadi ukuran 22.594 data dikali dengan 100 (5 ukuran window dan 20 ukuran orthogonal encoding). Pada percobaan kali ini model SVM digunakan dengan kernel Radial Basis Function dengan parameter Gamma = 0.1 dan C=1.5

Hasil Akhir

Ukuran Window	Classification Report			
5		precision	recall	f1-score	support
	0	0.58	0.54	0.56	1414
	1	0.63	0.76	0.69	2039
	2	0.55	0.38	0.45	1066
	micro avg	0.60	0.60	0.60	4519
	macro avg	0.58	0.56	0.56	4519
	weighted avg	0.59	0.60	0.59	4519
7		precision	recall	f1-score	support
	0	0.57	0.58	0.57	1374
	1	0.63	0.74	0.68	2060
	2	0.55	0.36	0.44	1085
	micro avg	0.60	0.60	0.60	4519
	macro avg	0.58	0.56	0.56	4519
	weighted avg	0.59	0.60	0.59	4519
9		precision	recall	f1-score	support
	0	0.61	0.59	0.60	1409
	1	0.63	0.74	0.68	2021
	2	0.55	0.39	0.46	1089
	micro avg	0.61	0.61	0.61	4519
	macro avg	0.60	0.58	0.58	4519
	weighted avg	0.61	0.61	0.60	4519

11 **(Terbaik)**		precision	recall	f1-score	support
	0	0.64	0.57	0.60	1478
	1	0.63	0.76	0.69	2009
	2	0.56	0.41	0.47	1032
	micro avg	0.62	0.62	0.62	4519
	macro avg	0.61	0.58	0.59	4519
	weighted avg	0.62	0.62	0.61	4519
13		precision	recall	f1-score	support
	0	0.63	0.59	0.61	1484
	1	0.63	0.75	0.69	1996
	2	0.56	0.42	0.48	1039
	micro avg	0.62	0.62	0.62	4519
	macro avg	0.61	0.59	0.59	4519
	weighted avg	0.62	0.62	0.61	4519
15		precision	recall	f1-score	support
	0	0.60	0.60	0.60	1403
	1	0.64	0.74	0.68	2062
	2	0.57	0.40	0.47	1054
	micro avg	0.62	0.62	0.62	4519
	macro avg	0.60	0.58	0.59	4519
	weighted avg	0.61	0.62	0.61	4519