گزارش پروژه ی درس محاسبات زیستی

استاد درس : دکتر حاجی پور

دانشجو : سعید منصور لکورج – 99102304

من در این پروژه به صورت همزمان از MATLAB و PYTHON استفاده کردم و مشخص کردم.

فاز 1:

ویژگی های استفاده شده:(Matlab)

در این جا طبق متن گزارش قرار گرفته شده من ویژگی ها را به دو بخش تقسیم کردم:

گروه اول:

1. واریانس
2. بیشترین مقدار index در هسیتوگرام با 20 بخش
3. Kurtosis
4. Skewness
5. فرم فاکتور
6. میانگین
7. میانه
8. ماکسیمم
9. entropy

گروه دوم (در فرکانس):

1. میانگین
2. میانه
3. Band power
4. Obw
5. PSD-band1
6. ...
7. ...
8. ...
9. ...
10. ...
11. PSD-band7

و با استفاده از رابطه ی زیر fisher score را برای هر یک از ویژگی های بالا محاسبه کردم (در هر 59 کانال) تا در مرحله ی بعد به کمک آن بتوانم ویژگی های برتر را انتخاب کنم. (آنهایی که fisher score بالا تری دارند)

$$J = \frac{|S_b|}{|S_w|} = \frac{|\mu_0 - \mu_1|^2 + |\mu_0 - \mu_2|^2}{\sigma_1^2 + \sigma_2^2}$$

(درباره ی هر یک از پارامتر ها در فایل پروژه به صورت کامل توضیح داده شده است)

انتخاب ویژگی ها:(Matlab)

دیدگاه 1:

در این دیدگاه من یک ماتریس 20×59 درست کردم که ماتریس ویژگی های برای هر کانال است (البته اینکار را در دو مقیاس که در بالا جدا کردم در نظر گرفتم) و اینطور در نظر گرفتم که بر اساس fisher matrix برای هر کانال مثلا 5 ویژگی در نظر گرفتم پس ماتریس ویژگی ها برای هر sample به صورت 5×59 در می آید پس ماتریس کلی به صورت 295×550 در می آید.

دیدگاه 2:

دیدگاه دیگر من این بود چون سوال ما 2 کلاسه میباشد مثلا انتخاب این همه ویژگی زیاد است و به خاطر نحوه ی انتخاب ویژگی ها توسط ما اصلا ممکن است که بعضی ویژگی های خوب را نگیریم و تعداد زیادی ویژگی بد بدست آورم (ممکن است بعضی از کانال ها اصلا نتوانند این دو کلاس را به خوبی از هم جدا کنند ولی باز ما از این کانال ها ویژگی بر میداریم)، پس ویژگی ها را به صورت کلی از ماتریس fisher بر میداریم بدون توجه به این که این ویژگی ها از کدوم کانال ها گرفته میشود پس در نهایت سایز ماتریس train ما به صورت 30×550 در می آید یعنی تعداد نورون های ورودی ما 20 می باشد.

من در این پروژه کد هر دو بخش را زدم اما کد **دیدگاه 1** به صورت کامنت می باشد و من سوال را به کمک دیدگاه دو حل کردم.

برای دیدگاه دوم با یک شبکه ی *MLP* ساده (با شرایط یکسان) برای مقادیر 12 تا 20 و 60 تا ویژگی شبکه را *train* کردم و بهترین تعداد ویژگی 60 بود.(تعداد ویژگی 95 فرقی در درصد دقت نداشت)

ماتریس باینری ویژگی ها در برنامه ی *matlab* قابل مشاهده است و چون به صورت کلی ویژگی ها را مشخص کردم (یک ماتریس باینری 20×59 که آنهایی که 1 هستند ویژگی های انتخاب شده می باشند) نام بردن هر ویژگی دشوار است و فقط چند مورد را به عنوان مثال نام می برم:

1. واریانس کانال 6
2. Kurtosis کانال 6
3. فرم فاکتور کانال 5
4. فرم فاکتور کانال 7
5. واریانس کانال 22
6. واریانس کانال 23
7. میانه ی کانال 50
8. میانه ی کانال 49
9. Skewness کانال 49
10. *PSR* امواج α کانال 2
11. *PSR* امواج δ کانال 5
12. *PSR* امواج δ کانال 6

...

تمام این ویژگی ها را میتوان از روی ***fisher matrix*** می توان گفت، برای مثال برای ویژگی های دوم ماتریس به صورت زیر است:

59x11 double

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	1	0	0	0	0	
3	0	0	0	0	0	0	0	0	1	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	
5	0	0	0	0	1	0	1	0	1	1	0	
6	0	0	0	0	1	0	0	1	1	1	0	
7	0	0	0	0	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	0	0	0	0	
9	0	0	0	0	0	0	0	0	0	0	0	
10	0	0	0	0	0	0	0	0	0	0	0	
11	0	0	0	0	0	0	0	0	0	0	0	
12	0	0	0	0	0	0	0	0	0	0	0	
13	0	0	0	0	1	0	0	0	0	1	0	
14	0	0	0	0	0	0	0	0	0	0	0	
15	0	0	0	0	0	0	0	0	0	0	0	
16	0	0	0	0	0	0	0	0	0	0	0	
17	0	0	0	0	1	0	0	0	0	1	0	
18	0	0	0	0	0	0	0	0	0	0	0	
19	0	0	0	0	0	0	0	0	0	0	0	
20	0	0	0	0	0	0	0	0	0	0	0	
21	0	0	0	0	0	0	0	0	0	0	0	
22	0	0	0	0	0	0	0	0	0	0	0	
23	1	1	0	0	1	0	1	1	1	1	0	
24	0	0	0	0	0	0	0	0	0	0	0	
25	0	0	0	0	0	0	0	0	0	0	0	
26	0	0	0	0	0	0	0	0	0	0	0	
27	0	0	0	0	0	0	0	0	0	0	0	
28	0	0	0	0	0	0	0	0	0	0	0	

طبقه بندی به کمک شبکه های عصبی:(Python)

:MLP

در این بخش پس از اینکه ماتریس ویژگی ها را در matlab تولید کردم آنرا در python وارد میکنم تا کلاس بندی را انجام دهم:

به کمک GridSearch هم 5-fold validation وهم انتخاب hyper parameter ها را انجام میدهیم:

```
param_grid = {
    'activation': ['identity', 'logistic', 'tanh', 'relu'],
    'hidden_layer_sizes': [(i,) for i in range(10, 31)],
    'max_iter': [1000],  # Number of epochs
}
```

بهترین نتیجه به صورت زیر می باشد:

```
Best Hyperparameters:  {'activation': 'tanh', 'hidden_layer_sizes': (13,),
'max_iter': 1200}
Best Accuracy: 84.00%
```

:RBF

این بخش را به کمک کتابخانه ی torch انجام داده و hyper parameter هایی که باید بهینه سازی کنیم به صورت زیر می باشد:

```
param_grid = {
    'radius': [0.0125,0.025,0.05,0.1, 0.5, 1.0],
    'hidden_dim': [15, 20,30, 40,50,60,70]
}
```

فاصله ی استفاده شده در لایه ی نهان فاصله ی اقلیدسی است و تابع فعال سازی نیز گوسی می باشد.

بهترین نتیجه:

```
Best Parameters: {'hidden_dim': 80, 'radius': 0.05}
Best Accuracy on Validation Set: 0.8581818181818182
```

:Phase2

الگوریتم ژنتیک:(Python)

این بخش از پروژه را هم به کمک کتابخانه ی deap را زدم.

ایده:

در ابتدا به کمک fisher score 600 ویژگی برتر را استخراج کردم و یک ماتریس 600×550 را از متلب به پایتون منتقل کردم. (توجه شود که تمامی بخش های استخراج ویژگی مانند این 600 تا یا 60 تا در مرحله ی قبل برای اینکه از کد های تکراری استفاده نشود با یک کد انجام شده است):

```
logic_feature1 = LogicFeatureSelection(matrix_fish1, 400);

feature3_1 = cat(3,var_f.',maxH.',kurt.', sk.', ratio.',avg.',med.',mx.',ent.');
featureM1_train = featureCreator(feature3_1, logic_feature1,400);
```

به عنوان مثال برای گروه اول ویژگی ها با تغییر عدد 400 به چیزی که میخواهیم ویژگی ها استخراج می شوند.

کدینگ:

یک کروموزوم 60 تایی که مقدار آلل ژن های آن اعداد متفاوت از 1 تا 1000 هستند.

تابع ارزیابی:

از معیار انتخاب ویژگی بر مبنای ماتریس پخشی (چند بعدی استفاده کردم)، یعنی تابع ارزیابی کروموزم را میگیرد که indices ویژگی های ما می باشد و با کمک دیتای اصلی و label ها این مقدار را بدست می آوریم:

$$S_1 = \frac{1}{N_1}\sum_{i\in C_1}(\mathbf{x}_i - \boldsymbol{\mu}_1)(\mathbf{x}_i - \boldsymbol{\mu}_1)^T$$
$$S_2 = \frac{1}{N_2}\sum_{i\in C_2}(\mathbf{x}_i - \boldsymbol{\mu}_2)(\mathbf{x}_i - \boldsymbol{\mu}_2)^T$$

$$S_W = S_1 + S_2$$

اصلاح:

قبل از اینکه به مرحله ی بعد برویم ژن های تکراری را با اعداد متفاوت که در کروموزم نیستند جایگزین می کنیم.

جهش:

از shuffle استفاده میکنیم.

کراس اور:

از کراس اور دو نقطه ای استفاده می کنیم.

نحوه ی انتخاب:

از تورنومنت استفاده می کنیم و در هر تورنومنت سه تا کروموزوم شرکت میکند.

نخبه گرایی:

در این مدل از نخبه گرایی هم استفاده می کنیم (0.1 از بهترین کروموزوم ها به نسل بعدی میروند بدون تغییر)

الگوریتم ژنتیک چرا کار ما را راحت تر می کند:
در واقع از یک همسایگی به تعداد $\binom{600}{60}$ که یک همسایگی بزرگ است و پیدا کردن بهترین مورد با انتخاب تک تک کاری بسیار سخت و طولانی میباشد را در چند دقیقه برای ما انجام میدهد.

طبقه بندی به کمک شبکه های عصبی:(Python)

این بخش دقیقا شبیه فاز یک می باشد و هیچ تغییری نکرده است، تنها ویژگی هایی که استفاده کردیم تغییر کرده است.

(ویژگی های انتخاب شده (بیشترین fitness را در ران کردن های متفاوت با ویژگی های اولیه ی متفاوت در الگوریتم ژنتیک) در فایل best_individual.npy ذخیره شده است.)

:MLP

نتیجه به صورت زیر می باشد:

```
Best Hyperparameters:  {'activation': 'tanh', 'hidden_layer_sizes': (25,),
'max_iter': 1000}
Best Accuracy: 87.09%
```

:RBF

نتیجه به صورت زیر می باشد:

```
Best Parameters: {'hidden_dim': 80, 'radius': 0.025}
Best Accuracy on Validation Set: 0.889090909090909
```

مشاهده می شود که دقت ها بهبود پیدا کرده است و دقت های خوبی نیز می باشد که نشان میدهد الگوریتم ما موفقیت آمیز بوده است.

خلاصه ی نتایج:

	MLP	RBF
Phase 1	0.84	**0.858**
Phase 2	0.87	**0.89**

نتایج تست:

فاز 1:

نتایج این بخش در فایل TestLabels_phase1.mat ذخیره شده است.(labelMLP و labelRBF)

فاز 2:

نتایج این بخش در فایل TestLabels_phase2.mat ذخیره شده است.(labelMLP و labelRBF)