به نام هستی بخش

گزارش تمرین سوم

درس شناسایی الگو

نویسنده: حمیدرضا ابوئی

شماره دانشجویی: ۴۰۲۶۱۷۵۰۹

استاد: دکتر دلیری

تاریخ: ۷ بهمن ۱۴۰۲

مقدمه

بردار ویژگی های استخراج شده از سه گروه مختلف از بیماری آلزایمر با استفاده از تئوری گراف، داده شده است. با استفاده از روش اعتبار سنجی Leave one out و تعمیم روش SVM به حالت سه کلاسه (با هر رویکردی که میتوانید)، طبقه بندی دادگان را انجام داده و کانفیوژن ماتریس را برای بهترین دقت بدست آمده رسم کنید (کانفیوژن ماتریس 3 در 3). از هر روش انتخاب ویژگی (فیشر، کراسکال والیس و ...) میتوانید استفاده کنید. (استفاده از چند روش انتخاب ویژگی مختلف و گزارش دقت های حاصل از هر کدام نمره اضافی دارد)

«دیتا مربوط به ویژگی های استخراج شده از سه گروهه

فرض کنید مثلا سه گروه شامل سالم، افسرده خفیف، افسرده ضعیف

برا هر گروه ۱۰ تا سابجکت داریم.

برای هر سابجکت ۲۶ تا مقدار آستانه تعریف شده. و در هر آستانه هم ۱۱۶ تا ناحیه مغزی داریم (حالا اینجوری هم میشه در نظر گرفت که ۱۱۶ تا ناحیه داریم و ۲۶ تا آستاته برای هر ناحیه تعریف شده، فرقی نمیکنه در کل)

برای هر کدوم از این ۱۱۶ تا ناحیه تقسیم بندی شده در ۲۶ آستانه، اگر ماتریس مربوطشو باز کنید ۹ مقدار ویژگی تعریف شده که هر کدوم از این ۹ تا، نماینده ی یکی از ویژگی های گرافه.»

در این پروژه، با توجه به توضیحات داده شده و بررسی دیتاست متوجه می‌شویم که گروه سوم دارای ۱۱ سابجکت می‌باشد.

در ابتدا پس از لود کردن داده‌ها، نیازمند این هستیم که همه داده‌های موجود را به یک ماتریس تبدیل کنیم بدین منظور، داده‌ها و ویژگی‌های استخراج شده را در کنار هم قرار می‌دهیم و سپس لیبل هر گروه را در یک متغیر دیگر تعیین می‌کنیم.

```
%% Read data
load('.\hw3_TA_data_pattern.mat');
%% Create matrix - gather all data into one single matrix
TH = 26; % number of thresholds
all_region_matrix_gp1 = [];
all_region_matrix_gp2 = [];
all_region_matrix_gp3 = [];


for region = 1:116
    region_matrix_gp1 = [];
    region_matrix_gp2 = [];
    region_matrix_gp3 = [];

    for q = 1:TH
        region_matrix_gp1 = [region_matrix_gp1; All_gp1_local_feature{1,region}{q}];
        region_matrix_gp2 = [region_matrix_gp2; All_gp2_local_feature{1,region}{q}];
        region_matrix_gp3 = [region_matrix_gp3; All_gp3_local_feature{1,region}{q}];
    end
```

```
    all_region_matrix_gp1 = [all_region_matrix_gp1 ;region_matrix_gp1];
    all_region_matrix_gp2 = [all_region_matrix_gp2 ;region_matrix_gp2];
    all_region_matrix_gp3 = [all_region_matrix_gp3 ;region_matrix_gp3];

end

All_features = [all_region_matrix_gp1, all_region_matrix_gp2,
all_region_matrix_gp3]';

num_sub_gp1 = 10;
num_sub_gp2 = 10;
num_sub_gp3 = 11;
sb = num_sub_gp1 + num_sub_gp2 + num_sub_gp3;

label = [ones(1,num_sub_gp1),ones(1,num_sub_gp2)*2,ones(1,num_sub_gp3)*3];
```

در این مرحله می‌دانیم که ۱۰ داده‌ی اول مربوط به گروه اول، ۱۰ داده دوم مربوط به گروه دوم و ۱۱ داده سوم مربوط به گروه سوم می‌باشد. بدین ترتیب برای عدم ایجاد بایاس اولیه، داده‌ها را shuffle می‌کنیم. بدین منظور از تابع randperm متلب استفاده می‌کنیم تا اندیس‌های رندوم شده را برایمان ایجاد کند.

```
rl = randperm(sb); % random label
data = All_features(rl,:);
label = label(rl);
```

در ادامه الگوریتم کراس ولیدیشن را با متد leave-one-out پیاده سازی می‌کنیم. بدین ترتیب تعداد foldها را با تعداد داده‌ها برابر می‌گذاریم از روش زیر استفاده می‌کنیم:

```
%% cross-validation
numfold = sb;
indices = crossvalind('kfold',sb,numfold); %leave-one-out
```

در ادامه، مشخص می‌کنیم که قصد داریم چه تعداد ویژگی برتر را در مرحله‌ی انتخاب ویژگی، انتخاب کنیم. در اینجا ما قصد داریم مقایسه کنیم بنابراین یک بازه به آن داده می‌شود:

```
feature_selection_range = 1:10;
```

پس از پیش تعریف برخی متغیرهای دیگر، وارد حلقه‌ی کراس ولیدیش می‌شویم و در هر بار حلقه مراحل زیر را دنبال می‌کنیم.
ابتدا داده‌های ترین و تست را جدا می‌کنیم.

```
correct_num = zeros(numfold,numel(feature_selection_range));
confusion_matrix= zeros(3,3,numel(feature_selection_range));
for i = 1:numfold
    % Seperate train and test
    X_train = data(indices~=i,:);
    Y_train = label(indices~=i);
    X_test  = data(indices==i,:);
    Y_test  = label(indices==i);
```

سپس قصد داریم که از دادگان آموزش استفاده کنیم و ویژگی‌ها را نرمالایز کنیم تا بایاسی برای ویژگی‌ها به وجود نیاید:

```
    % Normalization
    train_mean = mean(X_train);
```

```
train_std = std(X_train);
X_train = (X_train - train_mean)./train_std;
X_test =  (X_test - train_mean)./train_std;
```

در ادامه می‌توانیم الگوریتم‌های انتخاب ویژگی را اعمال کنیم. در اینجا از الگوریتم Chi2 بدین منظور استفاده شده که ترتیب اهمیت ویژگی‌ها را مشخص می‌کند. سپس برای راحتی کار، ترتیب ویژگی‌ها را در دیتاست اصلی مطابق الگوریتم Chi2 مرتب می‌کنیم.

```
ranking = fscchi2(X_train,Y_train);
X_train = X_train(:,ranking);
X_test = X_test(:,ranking);
```

در ادامه در یک حلقه، تعداد ویژگی‌های برتر مورد نظر را انتخاب می‌کنیم و دیتاست جدید را مشخص می‌کنیم:

```
for j = feature_selection_range
    X_train_fs = X_train(:,1:j);
    X_test_fs = X_test(:,1:j);
```

در این مرحله، از الگوریتم SVM برای طبقه‌بندی استفاده می‌کنیم. اما الگوریتم اصلی صرفا برای دو کلاس قابل استفاده می‌باشد. اما با استفاده از تعمیم آن با استفاده از روش one-vs-all سعی بر طبقه‌بندی صحیح داده‌ها داریم. بدین منظور سه بار الگوریتم SVM را بر روی داده‌هایمان اعمال می‌کنیم. هر سری یکی از کلاس‌ها به عنوان کلاس برگزیده و باقی کلاس‌ها در کلاس دوم قرار می‌گیرند. سپس از مدل آموزش دیده برای محاسبه‌ی خروجی مدل با ورودی تست استفاده می‌کنیم. بدین منظور، از predict استفاده می‌کنیم و از خروجی آن، score را نیز که نمایانگر likelihood تعلق ورودی به آن کلاس است می‌گیریم. دومین مقدار score مشخص کننده‌ی likelihood تعلق ورودی به کلاس دوم است. این مقادیر را جمع آوری کرده و در انتها، کلاسی که بیشترین score را داشته باشد به عنوان کلاس پیش‌بینی شده معرفی می‌شود.

```
% One-vs-all classification
SVMModels = cell(3,1);
Scores = zeros(1,3);
for k = 1:3
    Y_train_new = (Y_train==k);
    SVMModels{k} = fitcsvm(X_train_fs,Y_train_new);
    [a,score]=predict(SVMModels{k},X_test_fs);
    Scores(k) = score(:,2); % Second column contains positive-class scores
end
[~,Y_pred] = max(Scores,[],2);
```

در ادامه، برای محاسبه‌ی دقت، در صورت درست بودن پیش‌بینی، یک متغیر مشخص می‌کند که در انتها میانگین گیری خواهد شد.

```
correct_num(i,j) = correct_num(i,j) + (Y_pred==Y_test);
```

همچنین برای هر یک از تعداد ویژگی‌های برتر، یک confusion matrix محاسبه می‌شود:

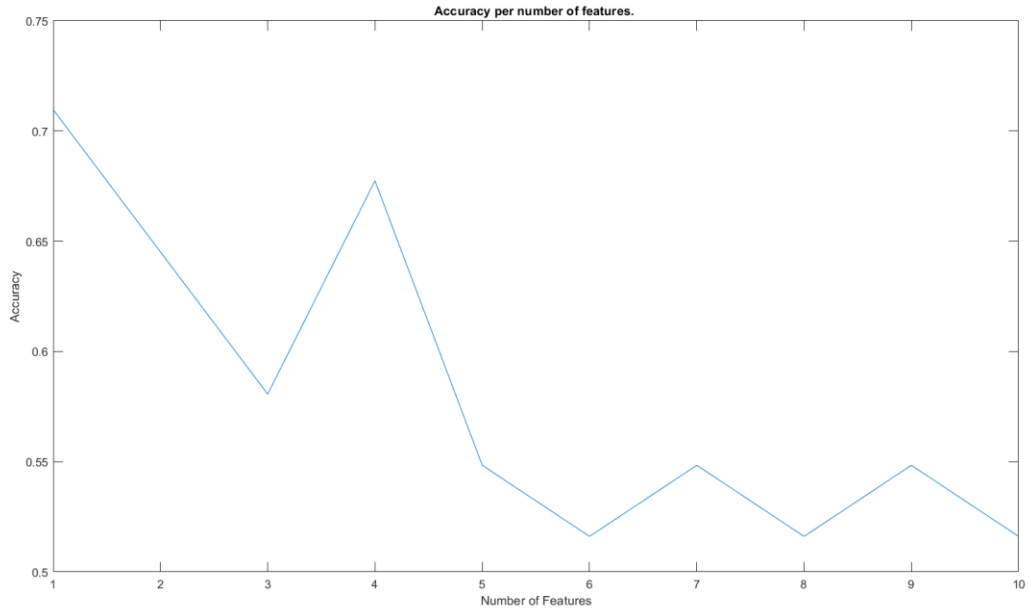
```
confusion_matrix(Y_test,Y_pred,j) = confusion_matrix(Y_test,Y_pred,j)+1;
```

محاسبه‌ی دقت و نمایش آن نیز پس از خارج شدن از تمام حلقه‌ها انجام می‌شود.

```
accs = mean(correct_num);

figure();
```

```
plot(feature_selection_range, accs)
xlabel("Number of Features");
ylabel("Accuracy");
title("Accuracy per number of features. ");
```



در انتها نیز، کانفیوژن ماتریس مربوط به بهترین دقت نمایش داده می‌شود:

```
[M,i] = max(accs);
fprintf("Best accuracy confusion matrix:\n");
disp(confusion_matrix(:,:,i));

output:
Best accuracy confusion matrix:
     6     1     3
     2     7     1
     0     2     9
```

همچنین برای انتخاب ویژگی می‌توان از روش‌های دیگری مانند فیشر استفاده کرد. بدین منظور از کتابخانه‌ی FSL استفاده می‌کنیم. متاسفانه در اینجا، به دلیل مشکلات خود کتابخانه امکان اجرا و نمایش نتایج وجود نداشت اما کد آن و تعلقات آن موجود می‌باشد:

```
%     ranking = spider_wrapper(X_train,Y_train,numF,lower('fisher'));
```

با تشکر.