



گزارش پیاده سازی مقاله :

**Optimizing the Channel Selection and Classification
Accuracy in EEG-Based BCI**

نام دانشجو:

محمد جواد سامری

استاد درس :

دکتر شالچیان

زمستان 96

پیش از پرداختن به تشریح پیاده سازی مقاله مطرح شده لازم است مرور مختصری را بر روی مقاله و دورنمای آن داشته باشیم . مقاله با این مقدمه ای بر متد های مورد استفاده قرار گرفته در زمینه انتخاب ویژگی شروع می کند و دلیل مطرح کردن ایده خود را این گونه مطرح می کند که در از آنجا که در انتخاب ویژگی با استفاده از الگوریتم های CSP و مشتقات آن ها برای انتخاب یک حد تعیین می شود و ویژگی هایی که مقدار ضریب آنها در فیلتر بدست آمده از آن حد بیشتر بود به عنوان ویژگی انتخاب می شوند , نویسنده نیز مطرح می کند که به دلیل اینکه کانال هایی که زیر حد مورد نظر بوده و حذف شده اند ممکن است حاوی اطلاعات مفیدی بوده باشند و حذف گردیده باشند لذا این روش به تنهایی روش بهینه ای نیست لذا مطرح می کند که در هنگام بدست آوردن فیلتر فضایی باید نگاهی هم به تنک بودن فیلتر نیز داشته باشیم.

سپس به سراغ تشریح الگوریتم CSP رفته آن را در غالب یک مسئله بهینه سازی با محدودیت مطرح می کند (شکل 1)

$$\min_{\mathbf{w}_i} \sum_{i=1}^{i=m} \mathbf{w}_i \mathbf{C}_2 \mathbf{w}_i^T + \sum_{i=m+1}^{i=2m} \mathbf{w}_i \mathbf{C}_1 \mathbf{w}_i^T$$

$$\text{Subject to : } \mathbf{w}_i (\mathbf{C}_1 + \mathbf{C}_2) \mathbf{w}_i^T = 1, \quad i = \{1, 2, \dots, 2m\}$$

$$\mathbf{w}_i (\mathbf{C}_1 + \mathbf{C}_2) \mathbf{w}_j^T = 0, \quad i, j = \{1, 2, \dots, 2m\} \quad i \neq j$$

شکل 1

پس از تشریح مسئله فوق مبحث تنک کردن CSP را در قالب مسئله بهینه سازی دیگری مطرح می کند (شکل 2)

$$\min_{\mathbf{w}_i} (1-r) \left(\sum_{i=1}^{i=m} \mathbf{w}_i \mathbf{C}_2 \mathbf{w}_i^T + \sum_{i=m+1}^{i=2m} \mathbf{w}_i \mathbf{C}_1 \mathbf{w}_i^T \right) + r \sum_{i=1}^{i=2m} \frac{\|\mathbf{w}_i\|_1}{\|\mathbf{w}_i\|_2}$$

$$\text{Subject to : } \mathbf{w}_i (\mathbf{C}_1 + \mathbf{C}_2) \mathbf{w}_i^T = 1 \quad i = \{1, 2, \dots, 2m\}$$

$$\mathbf{w}_i (\mathbf{C}_1 + \mathbf{C}_2) \mathbf{w}_j^T = 0 \quad i, j = \{1, 2, \dots, 2m\} \quad i \neq j \quad (11)$$

شکل 2

در حین تشریح کد های پیاده سازی شده مقاله با جزییات این مسئله بیشتر آشنا می شویم

پیاده سازی تابع هزینه

تابع هزینه مطرح شده در شکل 2 را پس از تبدیل به فرم ماتریسی در داخل فایل SCSPCostFunction.m پیاده سازی شده است که در زیر به تشریح این تابع می پردازیم

```
function [ Cost ] = SCSPCostFunction( W )
    % this function will compute the cost
    % assigned to each weight matrix
global C
    global R
    M = size(W,1) / size(C,2);
    Ccost = zeros(size(C,2),1);
    Cost = 0;
    WCost = 0;
    %% computin cost for C
    for i_class = 1 : size(C,2)
        AvgC = zeros(size(C{1}));
        for i_avgclass = 1 : size(C,2)
            if i_avgclass ~= i_class
                AvgC = AvgC + C{i_avgclass};
            end
        end
        AvgC = AvgC / (size(C,2)-1);

        for i_m = ((i_class-1)*M)+1 : (i_class*M)
            %disp(['first',num2str(i_m)])
            %size( (W(i_m,:)) )
            Ccost(i_class) = Ccost(i_class) + (W(i_m,:) * (
AvgC ) * W(i_m,:));
        end
    end
end
```

از آنجا که هدف از پیاده سازی الگوریتم جامعیت بخشیدن به آن و قابل استفاده بودن آن در تحقیقات دیگر است ابتدا در این تابع تعداد کلاس های داده ها را از روی ابعاد بدست آورده و سپس قسمت اول تابع هزینه که حاصل جمع ضرب سطریهای فیلتر بدست آمده تا این مرحله در ماتریس کواریانس است را برای هر کلاس محاسبه می کنیم که حاصل عددی صحیح بدست می آید.

```
%% computing cost for sparsity of weights
% each row has it's own penalty
Penalty = [];
for i_m = 1 : size(W,1)
    %computing l1 norm
    L1_Penalty = sum(sum(abs(W(i_m,:))));
    %computing l2 norm
    L2_Penalty = sqrt(sum(sum(power(W(i_m,:),2))));
    %computing sparsity term
```

```

        Penalty(i_m) = L1_Penalty / L2_Penalty ;
    end
    Penalty = sum(Penalty);
    %% sum of Costrs with respect to regularization value
    for i_class = 1 : size(C,2)
        % equation (11) on paper
        WCost = WCost + Ccost(i_class);
    End

```

سپس به سراغ محاسبه پناالتی ای می رویم که به ازای هر عنصر که 0 نباشد به مقدار هزینه ما اضافه می گردد این جریمه از تقسیم جریمه L1 بر جریمه L2 بدست می آید

```

    Cost = ((1-R) * WCost) + (R * Penalty);
end

```

و در قسمت آخر محاسبه مقدار تابع هزینه نهایی را از هزینه های بدست آمده را با توجه به ضریب رگوله سازی محاسبه می کنیم

پیاده سازی تابع محدودیت

```

function [ c, ceq ] = SCSPConstraint( W )
    global C ;
    c = [];
    SumC = zeros(size(C,1));
    for i_class = 1 : size(C,2)
        SumC = SumC + C{i_class};
    end
    ceq = W * (SumC) * W' - eye(size(W,1));
end

```

تابع محدودیت یک ورودی دارد که وزن های فیلتر به دست آمده تا کنون است و متغیر C که محدودیت های نا برابری است و زمانی مقدار می گیرد که ما در محدودیت مورد نظر گرفته شده نا مساوی داشته باشیم متغیر ceq مربوط به محدودیت های برابری است و معادله های محدودیت را هر کدام در یک سطر از این ماتریس قرار می دهیم نکته ای باید در هنگام استفاده از این متغیر در نظر داشت این است که تمامی معادله باید در یک سمت قرار گیرد

نکته دیگری که در مورد این تابع قابل ذکر است این است که در متغیر Ceq این محدودیت پیاده سازی شده است که در هنگام ضرب W در C هنگامی که هر دو W یکسان باشند جواب معادله برابر 1 است و در غیر این صورت برابر 0

است که این مهم با استفاده از یک ماتریس مربعی که فقط قطر آن 1 است و بقیه عناصر آن مقدار 0 دارند بر آورده شده است .

پیاده سازی حل مسئله بهینه

```
function [ Weights ] = SCSP_OPTIMIZER(
ClassTrain,M,Regularizer,
    MaxFunction,
    ToleranceCost,ConstTolerance,MaxItr)
if ~exist('MaxFunction','var')
    % if max function evaluation does not exist
    MaxFunction = 500000;
end
if ~exist('MaxItr','var')
    % if max function evaluation does not exist
    MaxItr = 10000;
end
if ~exist('ToleranceCost','var')
    % if max function evaluation does not exist
    ToleranceCost = 1e-7;
end
if ~exist('ConstTolerance','var')
    % if max function evaluation does not exist
    ConstTolerance = 1e-7;
end
if ~exist('M','var')
    % if max function evaluation does not exist
    M = 2;
end
if ~exist('Regularizer','var')
    % if max function evaluation does not exist
    Regularizer = 0;
end
global C;
global R;
R = Regularizer;
C = {};
```

در بخش اول این تابع به آماده سازی پارامتر هایی که که تابع بهینه ساز به آن نیاز پیدا خواهد کرد می پردازیم و طبق پارامتر هایی که مقاله معرفی کرده است مقدار دهی می کنیم

```

for i_class = 1 : size(ClassTrain,2)
    C{i_class} = zeros(size(ClassTrain{i_class}{1},2),
size(ClassTrain{i_class}{1},2));
    for i=1:size(ClassTrain{i_class},2)
        x = ClassTrain{i_class}{i}';
        for i_ch=1:size(x,1)
            x(i_ch)=x(i_ch)-mean(x(i_ch));
        end
        C{i_class} = C{i_class} + (x*x')/trace(x*x');
    end
end
for i_class = 1 : size(ClassTrain,2)
    C{i_class} = C{i_class} / size(ClassTrain{i_class},2);
end

```

در این قسمت به برای هر کلاس به ازای هر آزمایشی که انجام گرفته است ماتریس کواریانس داده های ثبت شده را محاسبه می کنیم و سپس برای نرمال کردن آنها بر مقادیر قطر تقسیم می کنیم و در آخر ماتریس کواریانس هر کلاس را با توجه به تعداد آزمایش ها میانگین می گیریم

```

options = optimoptions('fmincon',
'MaxFunctionEvaluations',MaxFunction,
'Display','off',
'FunctionTolerance',ToleranceCost,
'ConstraintTolerance',ConstTolerance,
'MaxIterations',MaxItr);

```

در این قسمت پارامترهای تعریف شده برای بهینه ساز را در قالب قرار می دهیم

```

notsatisfied = 1;

while(notsatisfied)
    w0 = rand(size(ClassTrain,2)*M,size(C{1},1));

    [Weights,fval,exitflag] = fmincon('SCSPCostFunction',
w0,
[],[],[],[],[],[],
'SCSPConstraint',options);
    if(exitflag ~=0)
        notsatisfied = 0;
    end
end

```

در این قسمت نیز بهینه ساز را فراخوانده ایم از آن جا که این الگوریتم نه کامل است به این معنی که اگر جوابی وجود داشته باشد الگوریتم تضمین می کند که جواب را پیدا کند و نه صحیح است به این معنی که اگر جوابی را پیدا کرد آن جواب لزوماً بهترین جواب نیست

لذا پس از هر بار پایان اجرای الگوریتم لازم است که بررسی شود که آیا محدودیت ها ارضا شده اند یا خیر که اگر ارضا نشده اند تابع بهینه ساز باید دوباره اجرا شود تا نهایتاً در این فضای جست و جوی بزرگ جوابی پیدا کنیم که محدودیت های ما را ارضا کند

پس از پیاده سازی ایده اصلی مقاله که الگوریتم SCSP بوده است ، به سراغ قسمت آزمایش و تحقیق رفته و مطابق با دستور العمل مقاله پیش می رویم

در مرحله اول که مربوط به فیلتر کردن و بخش بندی داده ها است . همان طور که در توضیحات مجموعه داده آمده است فرکانس نمونه برداری 250 هرتز است با استفاده از این نرخ نمونه برداری ضرایب فیلتری را در باند 8 تا 35 هرتز بدست می آوریم

```
[b_coef , a_coef]=butter(4,[8 35]/(Fs/2),'bandpass')
```

در قسمت بخش بندی کردن داده ها نیز طبق گفته مقاله داده های هر سیگنال از 0.5 ثانیه بعد از نمایش نشانگر تا 3.5 ثانیه بعد از آن قسمت مناسبی برای برش است و از آنجا که تحریک های مختلفی در طول آزمایش روی داده است قبل از اضافه کردن داده باید بررسی شود که آیا آزمایش معتبر است یا خیر که با بررسی اینکه آیا نوع آزمایش 1023 است یا خیر می توان از اعتبار آزمایش اطمینان حاصل کرد

```
validType = [769,770,771,772,783];  
  
if(ismember(Detail.EVENT.TYP(i_exp),validType))  
Signal{Counter} =  
Sig(Detail.EVENT.POS(i_exp)+(0.5*Fs):Detail.EVENT.POS(  
i_exp)+(3.5 * Fs),:);  
Signal{Counter} =  
filtfilt(b_coef,a_coef,Signal{Counter});  
Counter = Counter + 1;
```

End

متغیر `validType` حاوی پرچم های نشان دهنده این که نشانگر از کاربر چه عملی را خواسته که انجام دهد

سپس با توجه به برچسب واقعی داده ها آنها را در متغیر های مخصوص به خود قرار میدهیم

```
if(classlabel(i_trial) == 1)
class1{cnt1} = Signal{i_trial};
cnt1 = cnt1 + 1;
....
```

پس از قطعه بندی داده ها آنها را به دو دسته داده های آزمایشی و آموزشی تقسیم می کنیم

```
test_portion = 0.25;
[train , test] = crossvalind('HoldOut',
size(class1,2), test_portion);
testClass1      = class1(test);
trainClass1     = class1(train);
....
```

سپس همانطور که مقاله مطرح کرده است پس از تقسیم بندی الگوریتم **SCSP** را با $M=1$ بر روی داده های آموزشی با پارامتر های مختلف R و روش **10-fold** اجرا کرده و فیلتر ها را بدست می آوریم (فیلتر بدست آمده شامل 4 سطر است و 22 ستون است که هر ستون نشان دهنده یک کانال EEG است).

```
[ W_Sparse ] =
SCSP_OPTIMIZER({trainClass1,trainClass2,trainClass3,trainClass4},1,regularizer_coef(i_regularizer));
choosedIndex = find(abs(sum(W_Sparse)) > 10e-3);
SparseIndeces{i_regularizer} = choosedIndex;
```

اگر یا ستون تمام سطر های آن مقداری برابر صفر داشت یعنی آن ستون وابستگی آماری چندانی به تسک های ما ندارد و باید حذف شود . پس از حذف کانال های نامرتبط فیلتر **CSP** را بر روی داده ها با مقدار $M=3$ محاسبه می کنیم

پس از اجرای کامل روش **K-fold** و بدست آوردن R مناسب از روی داده ها با همان کانال های انتخاب شده مدلی را از فیلتر **CSP** عبور داده و با توان هر باند استخراج ویژگی را انجام می دهیم . و مدل **svm** چند کلاسی را بر روی داده های آموزشی آموزش می دهیم

```
TempSVM = templateSVM('KernelFunction','polynomial');
```



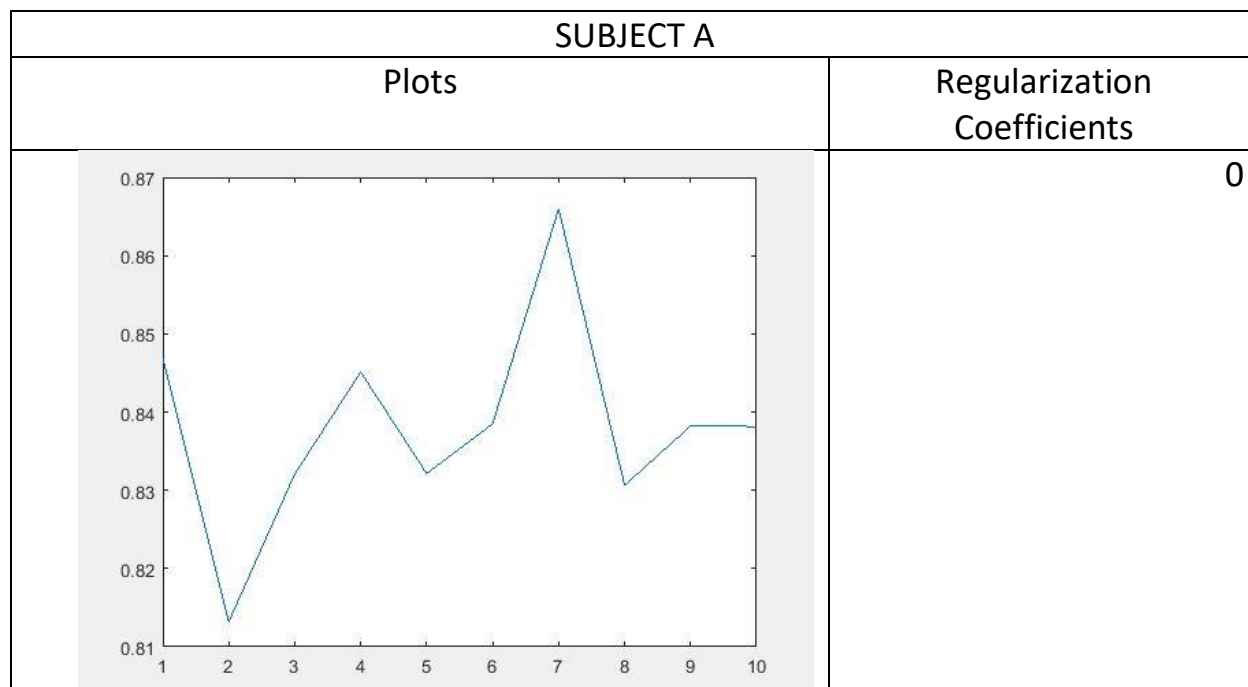
```
Model = fitcecoc(Ftrain,GroupTR,'Learners',TempSVM);
```

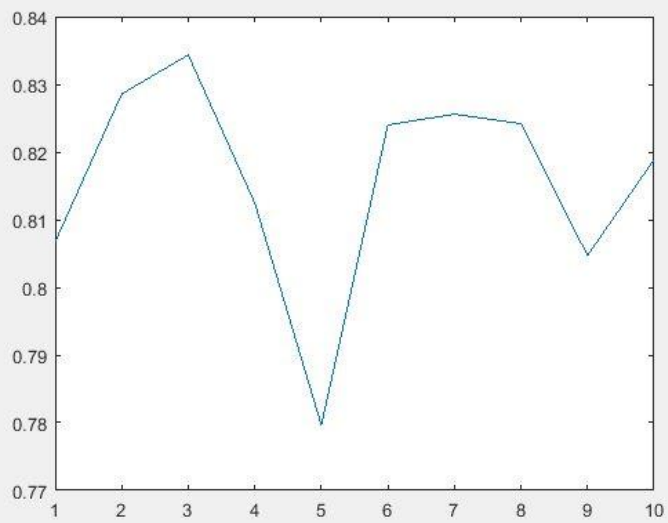
نکته ای که در مورد این قطعه قابل ذکر است این است که ابتدا یک قالب مدل SVM تعریف می کنیم و سپس با ارایه آن به تابع fitcecoc این تابع برای هر کلاس مدل SVM را مشتق می کند و آموزش می دهد

نتایج آزمایش پیاده سازی الگوریتم بر روی دو مجموعه داده

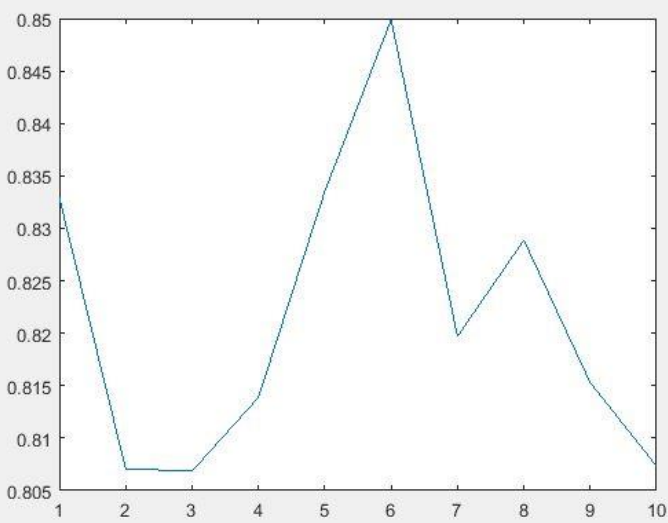
[BCI Competition IV 2b](#) _ 1

این مجموعه داده شامل 2 برچسب برای کلاس ها است و در مجموع از 9 فرد ثبت صورت گرفته است که نتایج مرحله 10-10-Fold را در نمودار ها مشاهده می کنید و نتایج الگوریتم با R انتخاب شده را بر روی داده های آزمایشی به دست می آوریم

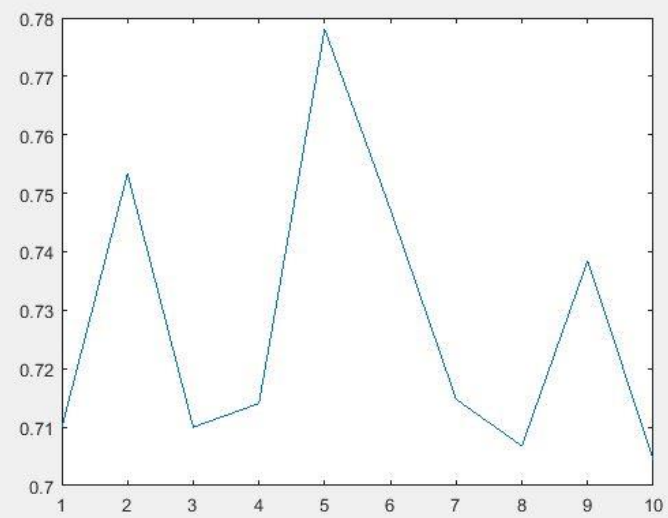




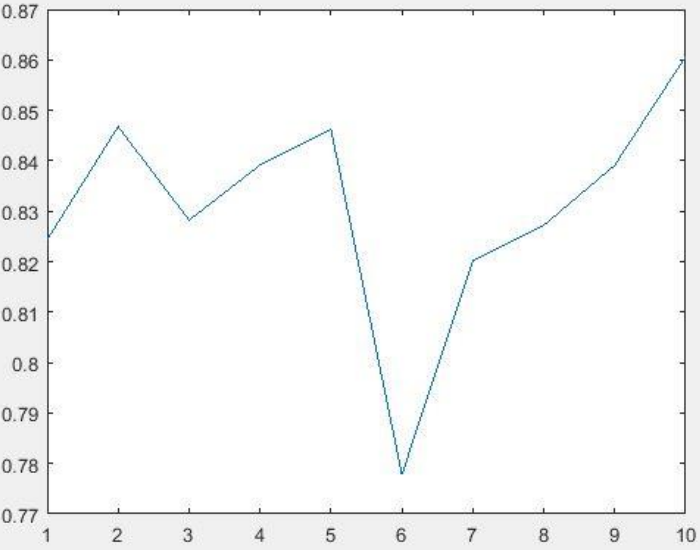
0.05

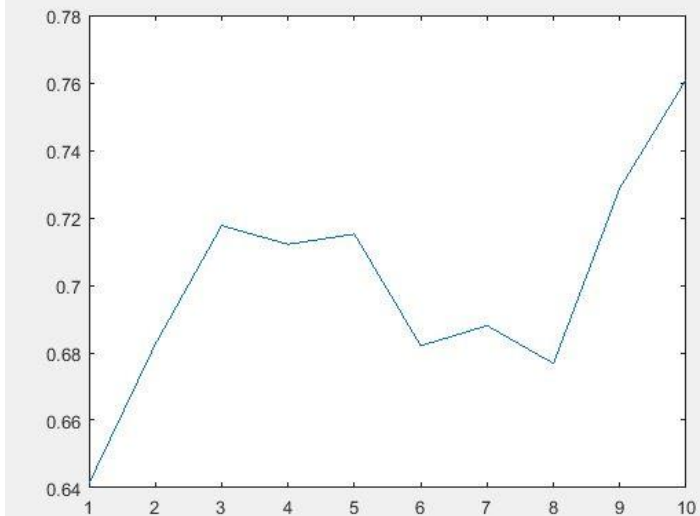


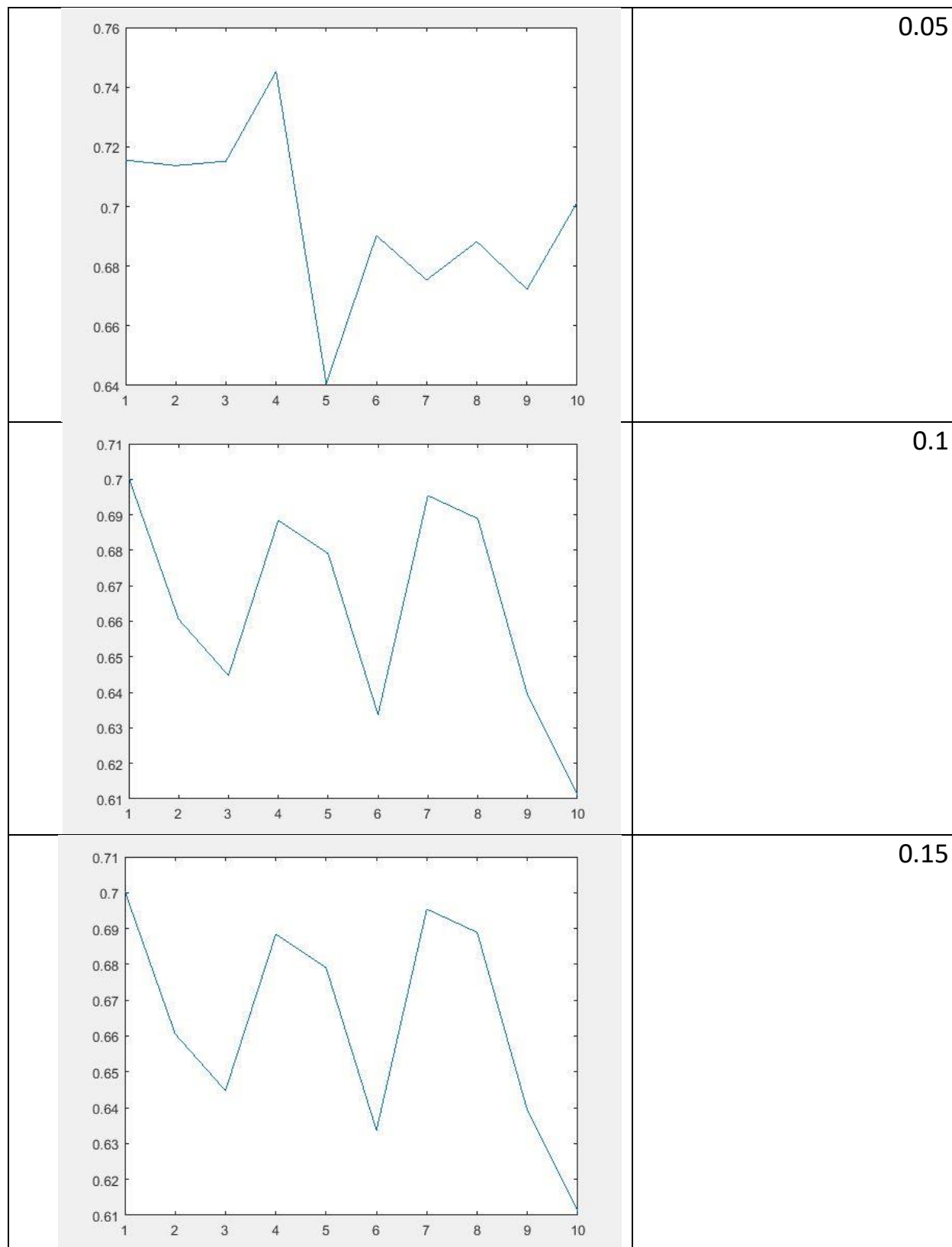
0.1

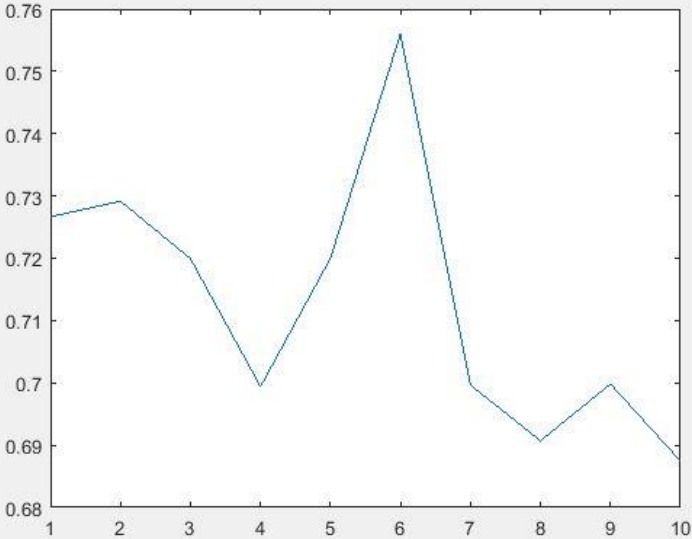


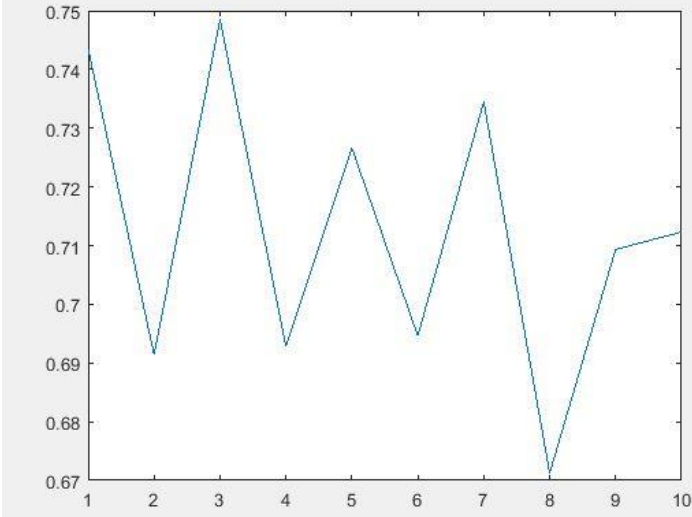
0.15

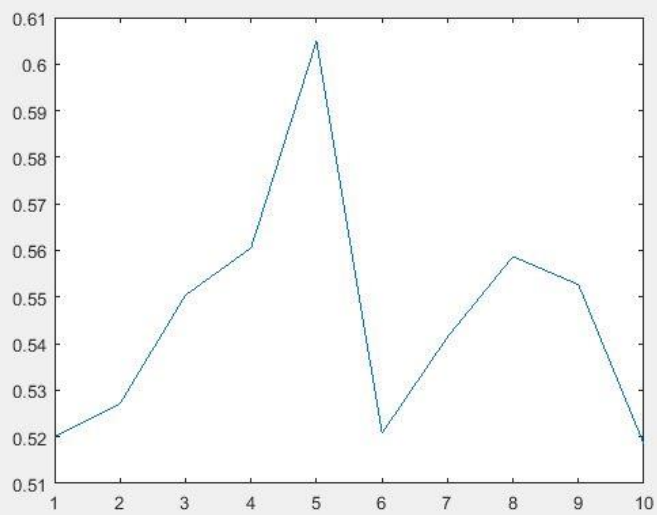
	0.2
0.7317	MEAN
0.1	SELECTED R
0.6934	Performance On Test data

SUBJECT B	
Plots	Regularization Coefficients
	0

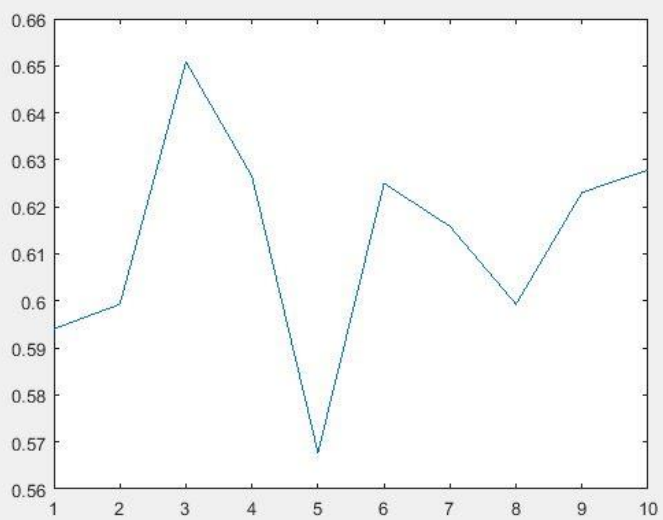


	0.2
0.67	MEAN
0.2	SELECTED R
0.8	Performance On Test data

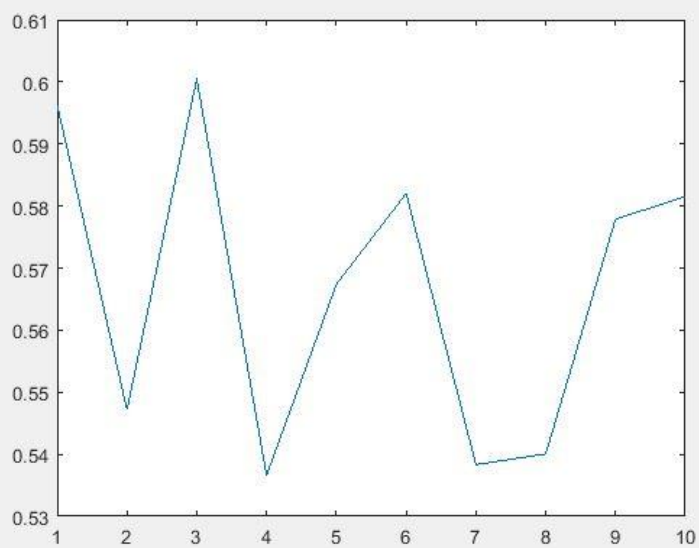
SUBJECT C	
Plots	Regularization Coefficients
	0



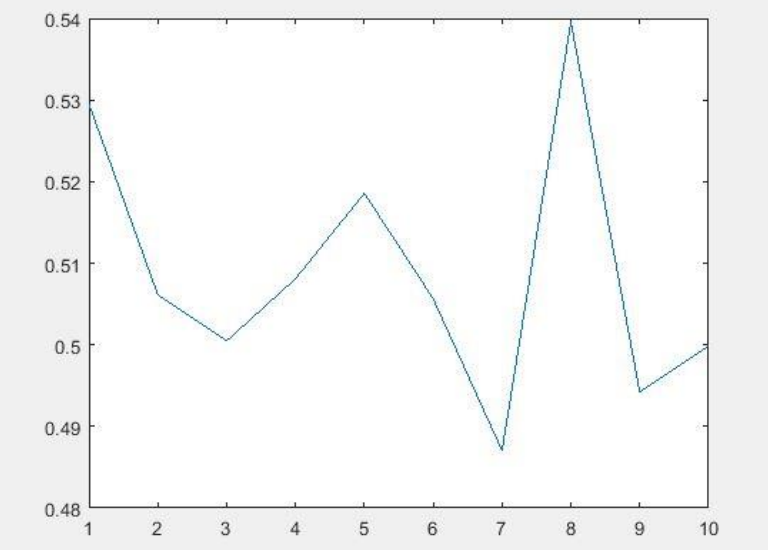
0.05

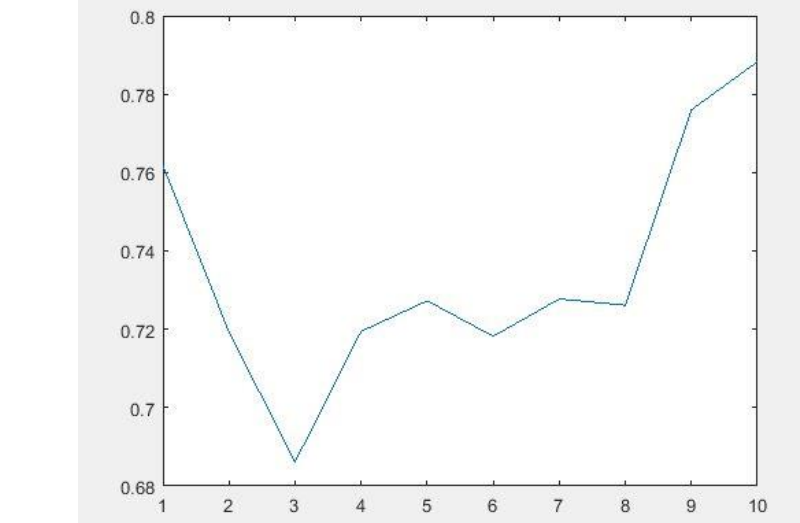


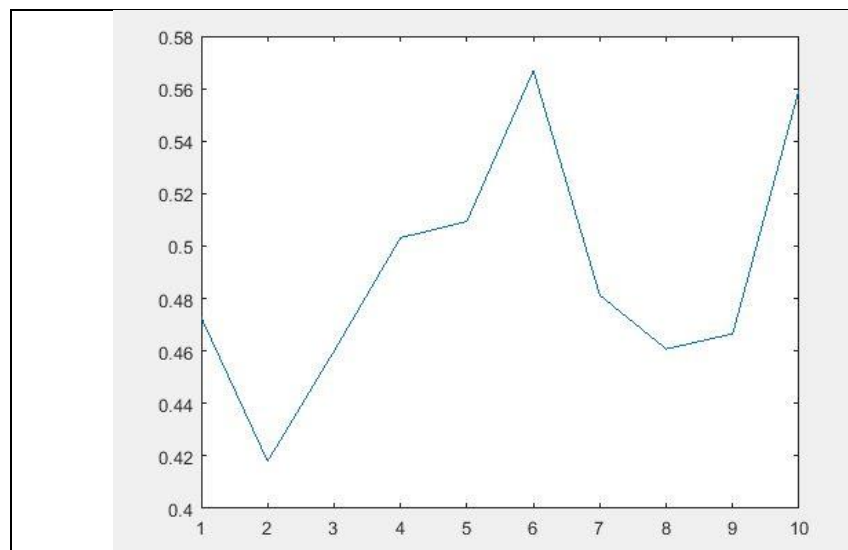
0.1



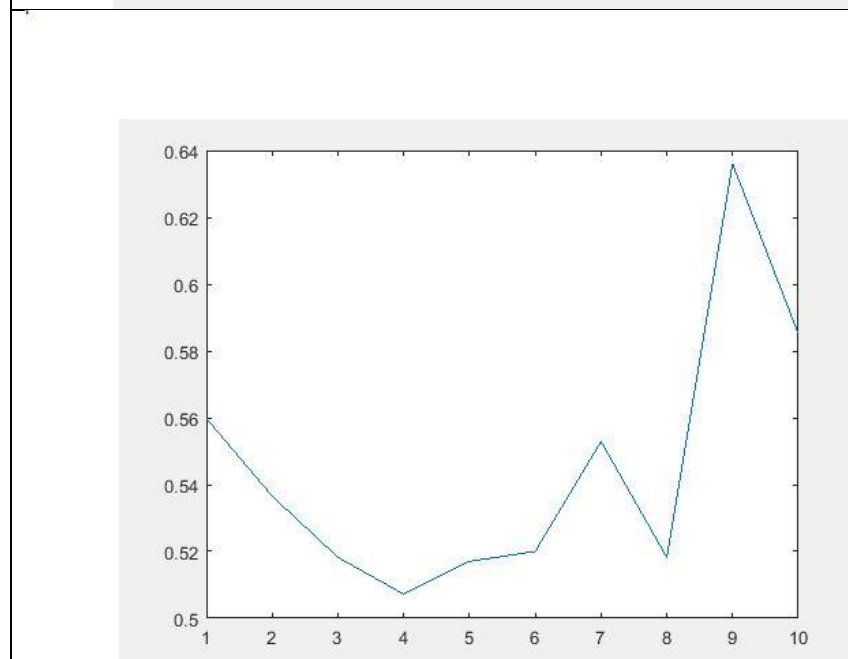
0.15

 <p>A line plot with x-axis from 1 to 10 and y-axis from 0.48 to 0.54. The data points are approximately: (1, 0.529), (2, 0.507), (3, 0.501), (4, 0.509), (5, 0.519), (6, 0.507), (7, 0.488), (8, 0.539), (9, 0.495), (10, 0.501).</p>	0.2
0.58	MEAN
0	SELECTED R
0.60	Performance On Test data

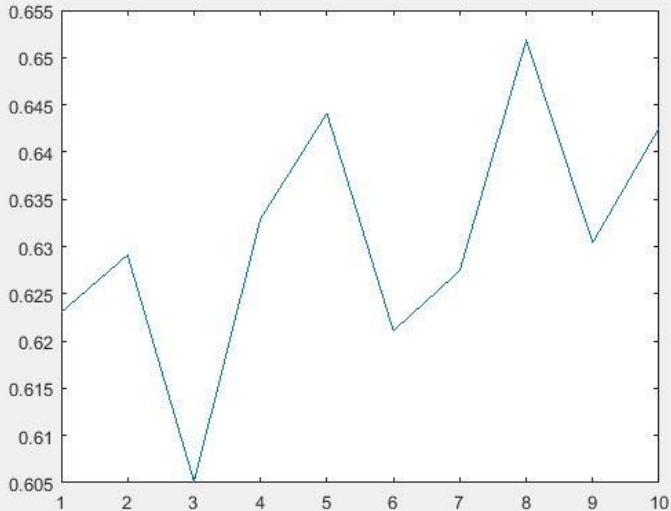
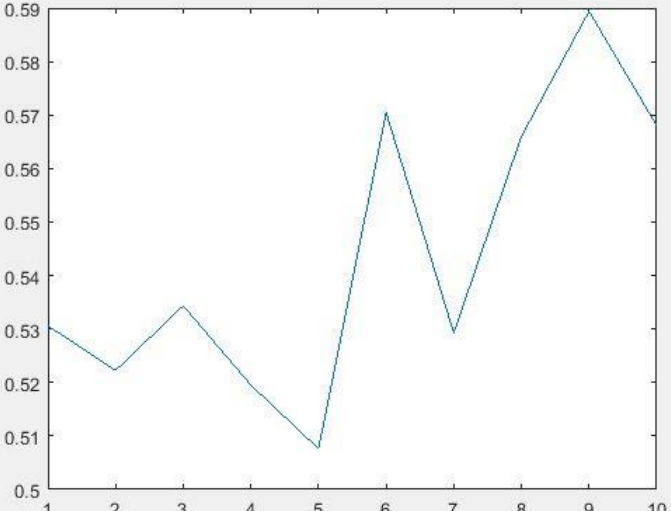
SUBJECT D	
Plots	Regularization Coefficients
 <p>A line plot with x-axis from 1 to 10 and y-axis from 0.68 to 0.8. The data points are approximately: (1, 0.761), (2, 0.721), (3, 0.688), (4, 0.721), (5, 0.728), (6, 0.719), (7, 0.728), (8, 0.727), (9, 0.777), (10, 0.781).</p>	0



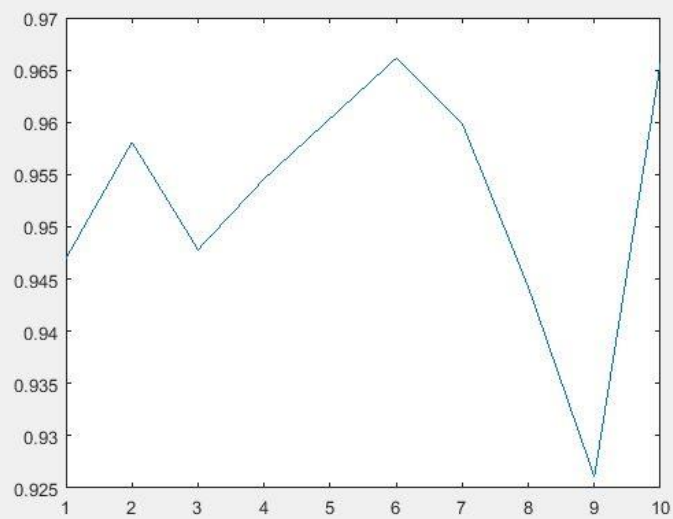
0.05



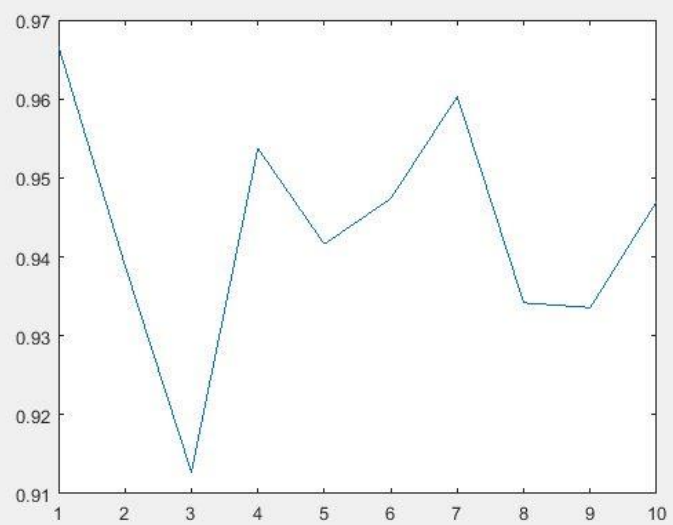
0.1

	0.15
	0.2
0.68	MEAN
0.1	SELECTED R
0.78	Performance On Test data

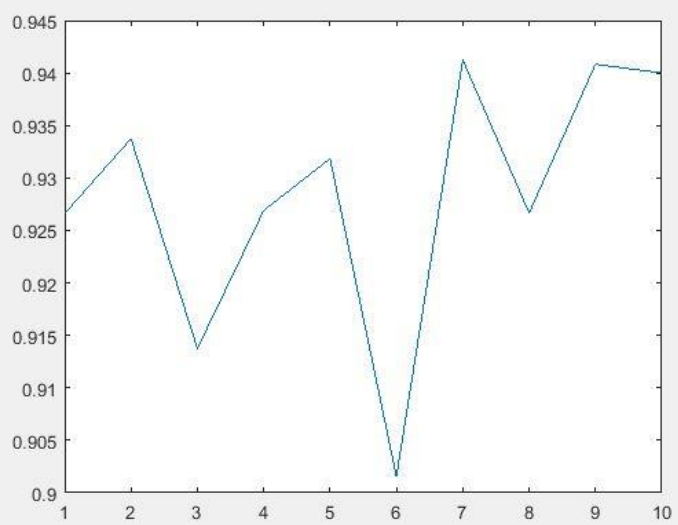
SUBJECT E	
Plots	Regularization Coefficients



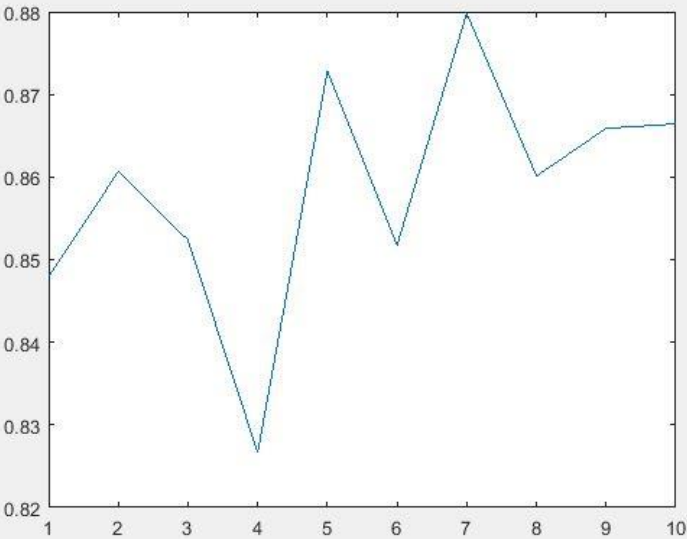
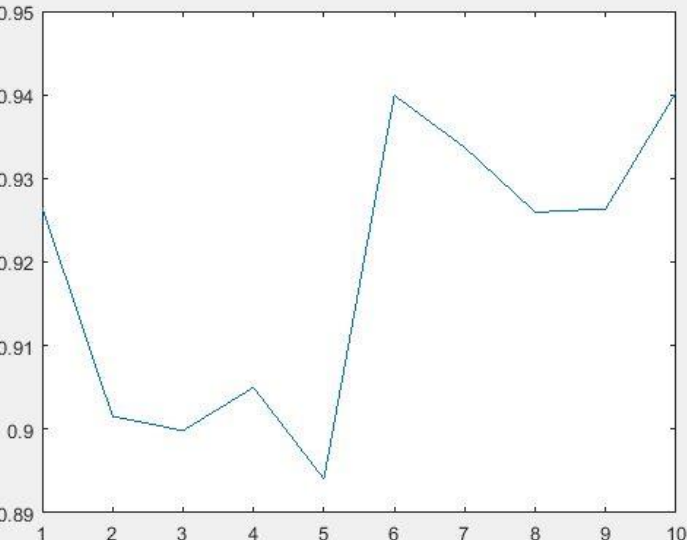
0



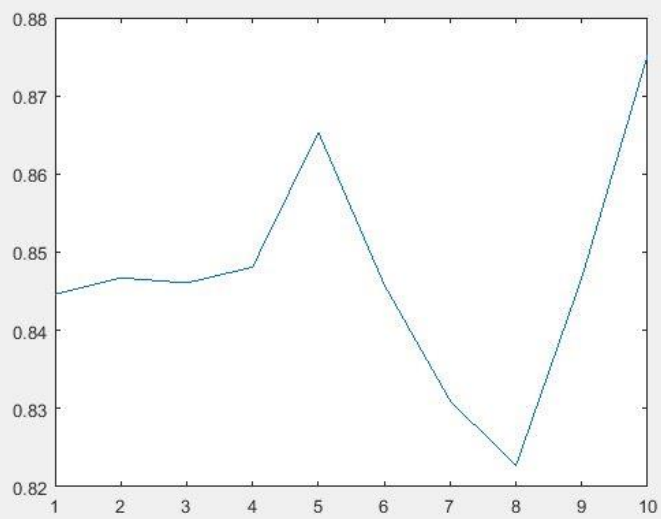
0.05



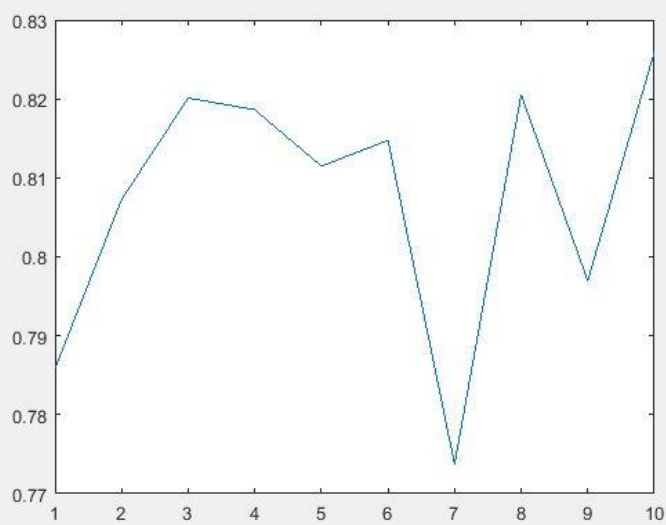
0.1

	0.15
	0.2
0.91	MEAN
0.1	SELECTED R
0.92	Performance On Test data

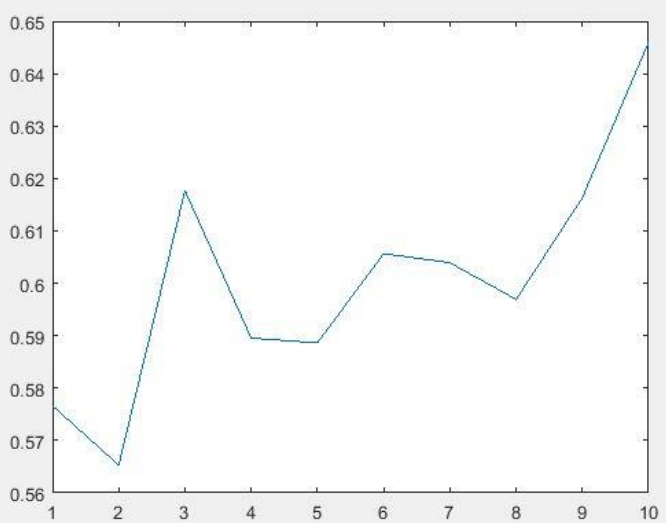
SUBJECT F	
Plots	Regularization Coefficients



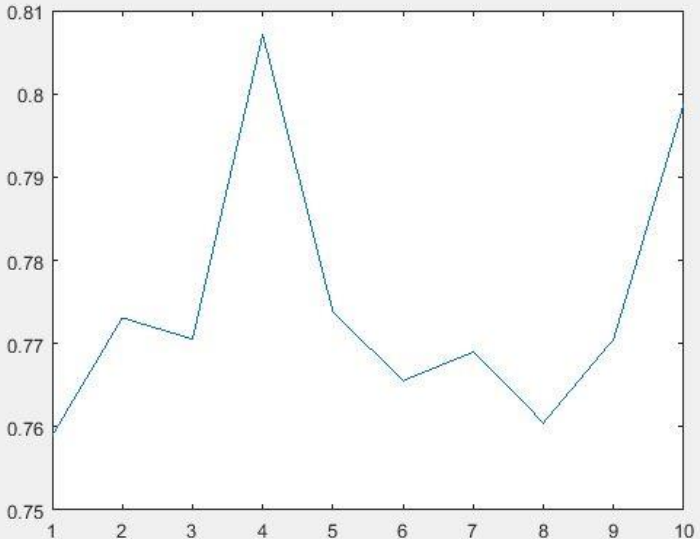
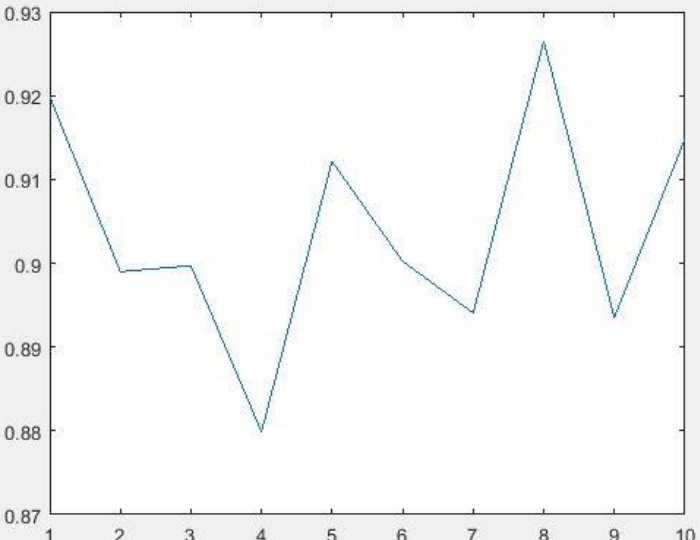
0



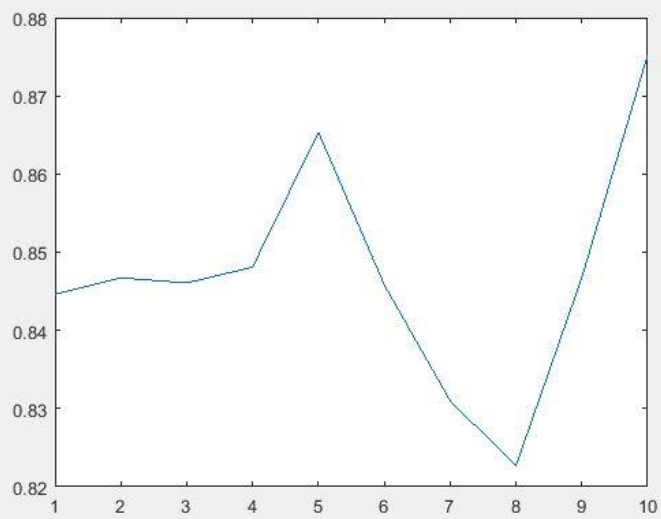
0.05



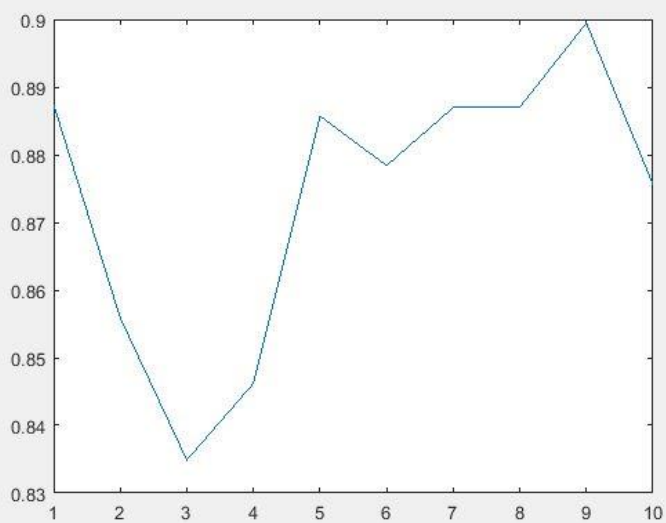
0.1

	0.15
	0.2
0.7912	MEAN
0.05	SELECTED R
0.90	Performance On Test data

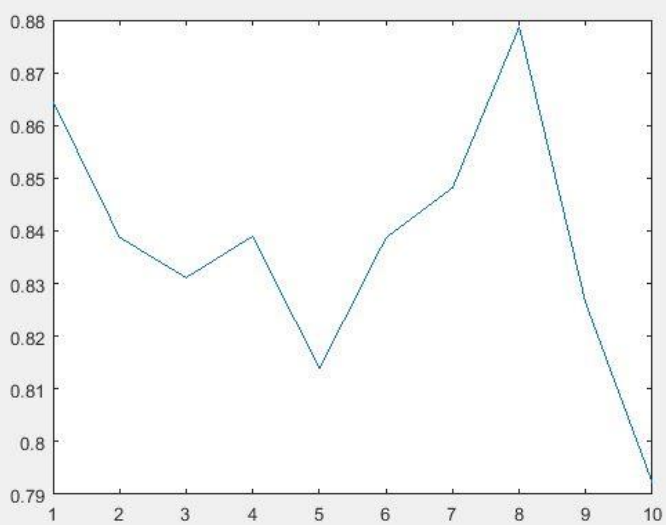
SUBJECT G	
Plots	Regularization Coefficients



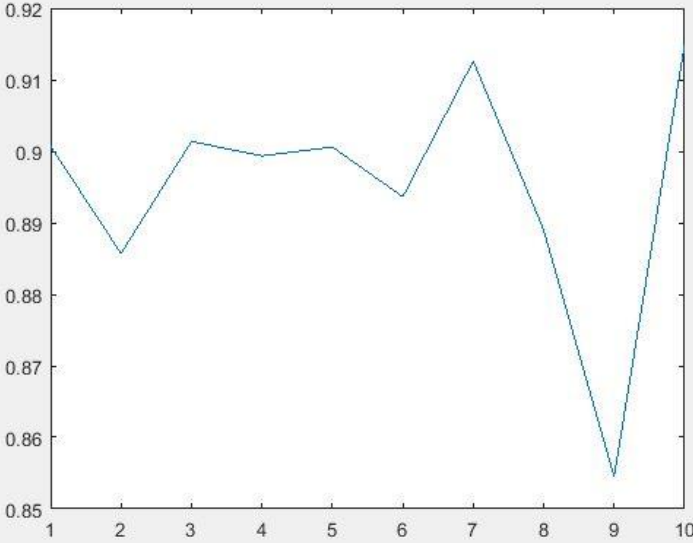
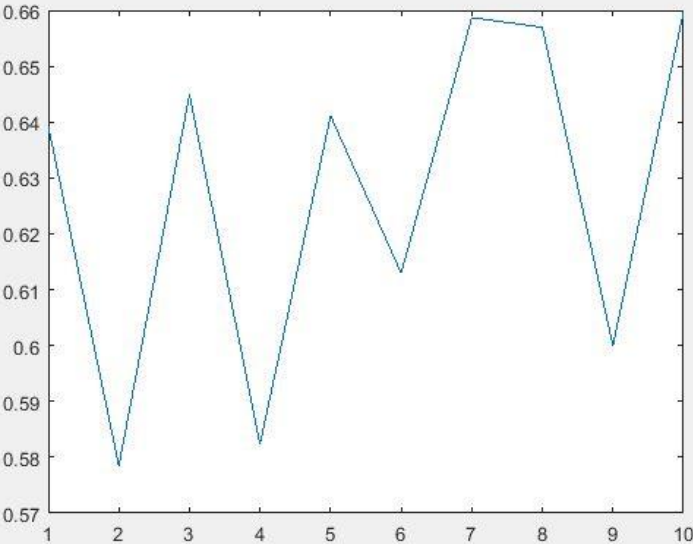
0



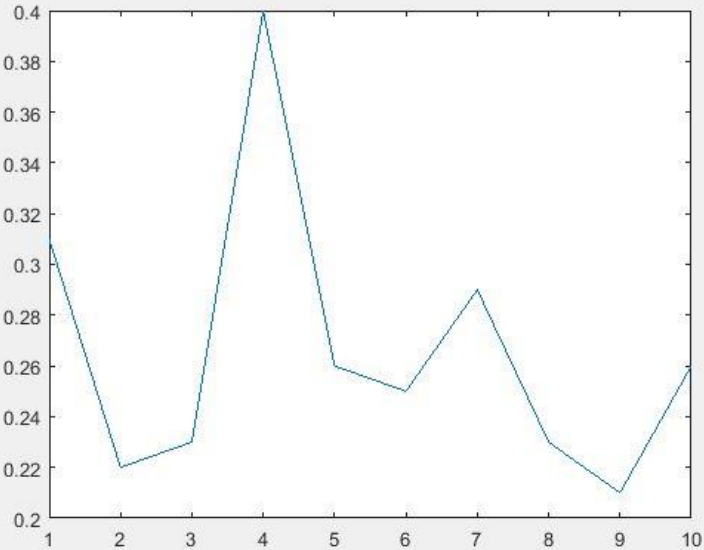
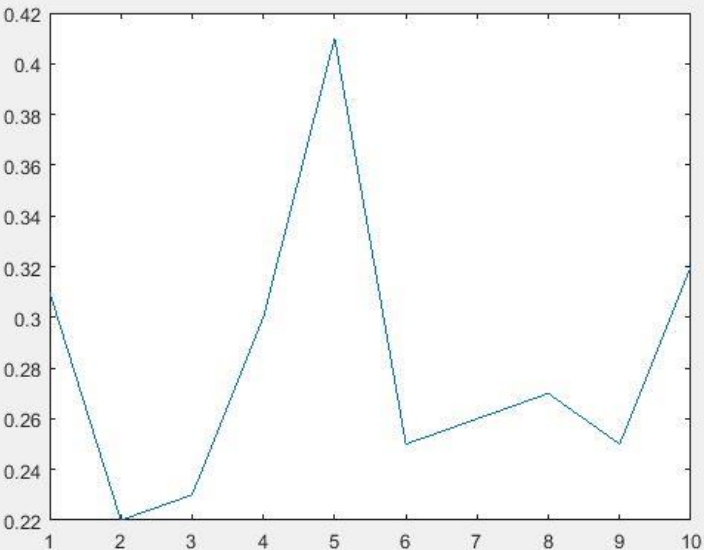
0.05



0.1

	0.15
	0.2
0.87	MEAN
0.15	SELECTED R
0.93	Performance On Test data

میانگین کارایی الگوریتم بالا بر روی نمونه های آزمایشی برابر 0.71429 است .

Plots	Regularization Coefficients
	0.0
	0.05
عملیات به صورت دستی متوقف شد	

از آنجا که طبقه بند کاملاً به صورت یک رده بند تصادفی عمل می کند عملیات را به صورت دستی متوقف کردیم . احتمال زیادی وجود دارد که منشا خطا آشنا نبودن اینجانب آشنا نبودن با این مجموعه داده و نحوه قطعه بندی آن است .