Xi He April 27, 2015
Due: May 1, 2015

1 Introduction

In this report, we test four classifiers on given data set II. We have four separate sets: A, B, C and D in data set II, which in total includes 10 uppercase character classes in a single type 'Times Roman'. Furthermore, we find a new feature and rebuild classifier which has the best performance among those four classifiers. Finally, we design a new classifier by SVM and compare the classified result with other classifier. Note that throughout all this report, we choose set A as training set and then using classifier to classify all sets A, B, C and D.

We use 20-dimension moments feature for the first four classifiers, and we state the 20 moment used in the implementation as following.

$$M_{i,j} \quad \text{where } i = 1, 2, \ldots, 6 \text{ and } j = i, \ldots, 7 - i, \text{ exclude } M_{6,1}. \tag{1}$$

By using this 20-dimension moments feature, we training data set A with the following four classifiers

M1. Moment-space minimum-distance classifier

M2. Moment-space classifier with identical covariances

M3. 1NN in moment space

M4. 5NN in moment space (breaking ties lexically).

After this, we choose new feature as the mean of pixel space in each position, i.e., in 256-dimension space. By making use of this feature on those four classifiers, we notice that the best performance comes from classifier $M3$ with 256-dimension pixel space, so we called this classifier as

M5. 1NN in pixed space.

Finally, we explore SVM classifier which based on the two-labels classifier function $svmtrain$ and $svmclassify$. Actually, what we did to use this method is test each two classes separately (which is 45 times compare for each sample), and then choose the most frequent correct answer to be the classified result. We also train and test this classifier in pixel space. We call this classifier as

M6. SVM in pixed space.

In section 2, we summary error table for each method when train on set A and test on set A, B, C and D respectively.

2 Summary Error Table

Classifier	A	B	C	D
M1	201	209	231	196
M2	39	39	49	37
M3	0	92	114	105
M4	128	143	158	164
M5	0	13	16	17
M6	0	8	9	4

Table 1: Error table for M1-M6 on data set II

We learn that in moment space, M2 shows the best classification result. Then, by definition, we derive the best average error rate is

$$E = \frac{39 + 49 + 37}{3} \approx 42. \tag{2}$$

One can see that both $M5$ and $M6$ cut E by at least a factor of two, which means these two methods satisfies requirement of this report.

3 Confusion Tables

In this section, we list all six confusion table of these six classifiers. Note that the confusion table is derived as training on set A and testing on entire data set II. In other word, each of the confusion table is the sum of four confusion table by using a classifier on training on set A and testing on A, B, C and D respectively.

	******** Method 1: Moment-space minimum-distance classifier *******										
Classified as:	B	C	D	E	I	J	O	R	U	V	ErrorTypeI
True class											
B	288		23		23	4	39	23			112
C		339		28	5	3			25		61
D	73		296		6	10	13	2			104
E	1			269	40	24	3	60	3		131
I					397	3					3
J					38	362					38
O	15				66	11	291	17			109
R	24		1	6	32	5	46	286			114
U		23		3	27	4			277	66	123
V	1				30				11	358	42
ErrorTypeII	114	23	24	37	267	64	101	102	39	66	837(Total Error)

Figure 1: Moment-space minimum-distance classifier

	**	Method 2: Moment-space classifier with identical covariances						***		
Classified as:	B	C	D	E	I	J	O	R	U	V	ErrorTypeI
True class											
B	385		9		1	3	1	1			15
C		395			5						5
D	38		346	1		2	13				54
E				378	16	4			2		22
I					398	2					2
J					28	372					28
O	5		2				393				7
R	5						1	394			6
U		3		5	10				382		18
V					3				1	396	4
ErrorTypeII	48	3	11	6	63	11	15	1	3		161(Total Error)

Figure 2: Moment-space classifier with identical covariances

	******	Method 3: 1NN in moment space	******							
Classified as:	B	C	D	E	I	J	O	R	U	V	ErrorTypeI
True class											
B	325		42			1	12	20			75
C		395		1	1				3		5
D	56		335				9				65
E	2	3		355	4	5	5	21	5		45
I					382	18					18
J					13	387					13
O	12		9	4			366	9			34
R	9		3	16	1		6	365			35
U		10		4	1		1		380	4	20
V							1			399	1
ErrorTypeII	79	13	54	25	20	24	34	50	8	4	311(Total Error)

Figure 3: 1NN in moment space

	******	Method 4: 5NN in moment space	******							
Classified as:	B	C	D	E	I	J	O	R	U	V	ErrorTypeI
True class											
B	383		7	1	2	1	4	2			17
C		398		1	1						2
D	186		211				3				189
E	9	12		365	6	2	1	3	2		35
I				1	398	1					2
J					42	358					42
O	59		12	28	6		295				105
R	64		2	47	2	2	23	260			140
U	1	35		14	1		1		348		52
V	5						1		3	391	9
ErrorTypeII	324	47	21	92	60	6	33	5	5		593(Total Error)

Figure 4: 5NN in moment space (breaking ties lexically)

```
                ******************    Method 5: 1NN in pixel space     ******************
Classified as:  B       C       D       E       I       J       O       R       U       V       ErrorTypeI
True class
        B       396                                                     4                       4
        C               399                     1                                               1
        D                       397                             3                               3
        E               1               397     2                                               3
        I                                       397     3                                       3
        J                                       21      379                                     21
        O                       2                               398                             2
        R       1                                                       399                     1
        U               5               2       1                               392             8
        V                                                                               400
ErrorTypeII     1       6       2       2       25      3       3       4                       46(Total Error)
```

Figure 5: 1NN in pixed space

```
                ******************    Method 6: SVM in pixel space     ******************
Classified as:  B       C       D       E       I       J       O       R       U       V       ErrorTypeI
True class
        B       398                                                     2                       2
        C               399                     1                                               1
        D                       399                             1                               1
        E                               394     1       1                       4               6
        I                                       395     5                                       5
        J                                       5       395                                     5
        O                                                       400
        R                                                               400
        U                                       1                               399             1
        V                                                                               400
ErrorTypeII                                     8       6       1       2       4               21(Total Error)
```

Figure 6: SVM in pixed space

4 Discussion

From error table in section 2 and confusion table in section 3, we can say that it's better to apply pixel feature and svm method to attain a relative low error rate. However, we have the following execution time for each classifier.

Classifier	M1	M2	M3	M4	M5	M6
CPUtime(s)	0.1718	0.7498	9.4580	9.4787	11.9983	84.2689

Table 2: Time cost for M1-M6 on data set II

From table 2, we may see that M5 and M6, who have relatively high classification accuracy, need more computation to obtain the result. And roughly speaking, nearest neighborhood methods are more costly since it need do more comparison and multiplication. With respect to M6, which is based on SVM method, it costly much more time. However, it is reasonable since to derive the classified result, we should run 45 times classification process, which is costly.

5 Program Listings

This section includes all the implementation details for this project. All the results can be computed by these program.

Listing 1: RESULT.m

```
Input_data;

deal_data_moment;

tic
classify_first;
toc

tic
classify_second;
toc

tic
classify_third;
toc

tic
classify_fourth;
toc

deal_data_pixel;

tic
classify_fifth_pixel;
toc

tic
classify_sixth_pixel;
toc
```

Listing 2: Input_data.m

```
f = dir('./C-II');

% Define constants
params.sample_Num = 4;
params.class_Num = 10;
params.info_Num = 100;
params.image_Size = 16;

data = struct();
```

```
global sample_Set;
sample_Set = ['A','B','C','D'];
global class_Set;
class_Set = ['B','C','D','E','I','J','O','R','U','V'];


for i = 3:length(f)
    name = f(i).name;
    class_Index = find(class_Set == name(3));
    sample_Index = find(sample_Set == name(1));
    data(class_Index).sample(sample_Index).name = name;
end

for j = 1:params.class_Num
    for k = 1:params.sample_Num

        sample_Data = ['./C-II/',data(j).sample(k).name];
        fid = fopen(sample_Data);

        tline = fgetl(fid);
        t = 1;
        p = struct();

        while ischar(tline)
            if tline ~= -1 & tline(1) == 'C'
                h = strfind(tline, 'h');
                w = strfind(tline, 'w');
                b = strfind(tline, 'b');
                m = str2num(tline(h+1:w-1));
                n = str2num(tline(w+1:b-1));

                MAT = char(ones(params.image_Size,params.image_Size)*'.');
                toprow = floor((params.image_Size-m)/2)+1;
                bottomrow = toprow + m - 1;
                leftcolumn = floor((params.image_Size-n)/2)+1;
                rightcolumn = leftcolumn + n - 1;
                MAT(toprow:bottomrow,leftcolumn:rightcolumn) =
                    reshape(fscanf(fid,'%s',[m,1]),n,m)';

                p(t).num = t;
                p(t).image = floor(double(MAT)/120);
                p(t).dimension = [m,n];
                t = t+1;
            end
            tline = fgetl(fid);
        end
        fclose('all');
```

```
        data(j).sample(k).info = p;
    end
end

clearvars -except data params
```

Listing 3: deal_data_moment.m

```
for i = 1: params.class_Num
    for j = 1:params.sample_Num
        for k = 1:params.info_Num
            MAT = data(i).sample(j).info(k).image;
            m_00 = sum(MAT(:));
            a = sum(MAT);
            b = sum(MAT');

            m_01 = 0;
            for p = 1:params.image_Size
                m_01 = m_01 + p*a(p);
            end

            m_10 = 0;
            for p = 1:params.image_Size
                m_10 = m_10 + p*b(p);
            end

            x_c = m_10/m_00;
            y_c = m_01/m_00;

            M = [];
            index = 1;
            for ii = 1:6
                for jj = 1:7-ii
                    M(index) = cen_mon(ii,jj,x_c,y_c,MAT,params);
                    index = index + 1;
                end
            end
            data(i).sample(j).info(k).cen_mon = M(1:end-1);
        end
    end
end

res = zeros(1,20);
for i = 1: params.class_Num
    for k = 1:params.info_Num
        res = res + data(i).sample(1).info(k).cen_mon.^2;
    end
end
```

```
rms = (res/1000).^0.5;

for i = 1: params.class_Num
    for j = 1:params.sample_Num
        for k = 1:params.info_Num
            data(i).sample(j).info(k).cen_mon_norm =
                data(i).sample(j).info(k).cen_mon./rms;
        end
    end
end

clearvars -except data params
```

Listing 4: classify_first.m

```
global sample_Set;
global class_Set;

train_mean = cell(params.class_Num,1);
for i = 1:params.class_Num
    res = zeros(1,20);
    for j = 1:params.info_Num
        res = res + data(i).sample(1).info(j).cen_mon_norm;
    end
    train_mean{i,1} = res/params.info_Num;
end

first_error = zeros(params.class_Num+1,params.class_Num+1);

for i = 1:params.sample_Num
    first_error_test = zeros(params.class_Num+1,params.class_Num+1);
    for j = 1:params.class_Num
        for k = 1:params.info_Num
            dist = [];
            for jj = 1:params.class_Num
                dist(jj) =
                    norm(train_mean{jj,1}-data(j).sample(i).info(k).cen_mon_norm,2);
            end
            min_dist = min(dist);
            index = find(dist == min_dist);
            first_error_test(j,index) = first_error_test(j,index) + 1;
        end
    end

    sum_type = sum(first_error_test(1:params.class_Num,1:params.class_Num));
    for p = 1:params.class_Num
        first_error_test(p,params.class_Num+1) =
            params.info_Num-first_error_test(p,p);
```

```
        first_error_test(params.class_Num+1,p) = sum_type(p) -
            first_error_test(p,p);
    end

    sum_total = sum(first_error_test');
    first_error_test(params.class_Num+1,params.class_Num+1) =
        sum_total(params.class_Num+1);

    first_error = first_error + first_error_test;
    fprintf('%d\t',sum_total(params.class_Num+1));
end

fprintf('\n\n\n\n             ********   Method 1: Moment-space minimum-distance
    classifier *******\n');
print(first_error)

clearvars -except data params
```

Listing 5: classify_second.m

```
global sample_Set;

train_mean = cell(20,1);
coviance_class = zeros(20,20);
for i = 1:params.class_Num
    res = zeros(1,20);
    Mat = zeros(params.info_Num,20);
    for j = 1:params.info_Num
        Mat(j,:) = data(i).sample(1).info(j).cen_mon_norm;
        res = res + data(i).sample(1).info(j).cen_mon_norm;
    end
    train_mean{i,1} = res/params.info_Num;
    coviance_class = coviance_class + cov(Mat);
end

sigma = coviance_class/params.class_Num;

second_error = zeros(params.class_Num+1,params.class_Num+1);

for i = 1:params.sample_Num
    second_error_test = zeros(params.class_Num+1,params.class_Num+1);
    for j = 1:params.class_Num
        for k = 1:params.info_Num
            dist = [];
            for jj = 1:params.class_Num
                distance(jj) =
                    0.5*(data(j).sample(i).info(k).cen_mon_norm-train_mean{jj,1})
                    ...
```

```
                / sigma *
                    (data(j).sample(i).info(k).cen_mon_norm-train_mean{jj,1})';
            end
            min_dist = min(distance);
            index = find(distance == min_dist);
            second_error_test(j,index) = second_error_test(j,index) + 1;
        end
    end

    sum_type = sum(second_error_test(1:params.class_Num,1:params.class_Num));
    for p = 1:params.class_Num
        second_error_test(p,params.class_Num+1) =
            params.info_Num-second_error_test(p,p);
        second_error_test(params.class_Num+1,p) = sum_type(p) -
            second_error_test(p,p);
    end

    sum_total = sum(second_error_test');
    second_error_test(params.class_Num+1,params.class_Num+1) =
        sum_total(params.class_Num+1);

    second_error = second_error + second_error_test;
    fprintf('%d\t',sum_total(params.class_Num+1));
end

fprintf('\n\n\n\n             **    Method 2: Moment-space classifier with
    identical covariances ***\n');
print(second_error)

clearvars -except data params
```

Listing 6: classify_third.m

```
global sample_Set;
global class_Set;

third_error = zeros(params.class_Num+1,params.class_Num+1);

for i = 1:params.sample_Num
    third_error_test = zeros(params.class_Num+1,params.class_Num+1);
    for j = 1:params.class_Num
        for k = 1:params.info_Num
            dist = [];
            for jj = 1:params.class_Num
                for kk = 1:params.info_Num
                    dist(jj,kk) = norm(data(j).sample(i).info(k).cen_mon_norm-...
                                    data(jj).sample(1).info(kk).cen_mon_norm,2);
                end
```

```
            end
            min_dist = min(min(dist));
            [index, datasample] = find(dist == min_dist);
            index = min(index);
            third_error_test(j,index) = third_error_test(j,index) + 1;
        end
    end

    sum_type = sum(third_error_test(1:params.class_Num,1:params.class_Num));
    for p = 1:params.class_Num
        third_error_test(p,params.class_Num+1) =
            params.info_Num-third_error_test(p,p);
        third_error_test(params.class_Num+1,p) = sum_type(p) -
            third_error_test(p,p);
    end

    sum_total = sum(third_error_test');
    third_error_test(params.class_Num+1,params.class_Num+1) =
        sum_total(params.class_Num+1);

    third_error = third_error + third_error_test;
    fprintf('%d\t',sum_total(params.class_Num+1));
end

 fprintf('\n\n\n\n             ******************  Method 3: 1NN in moment space
     ******************\n');
 print(third_error)

clearvars -except data params
```

Listing 7: classify_fourth.m

```
global sample_Set;
global class_Set;

fourth_error = zeros(params.class_Num+1,params.class_Num+1);

for i = 1:params.sample_Num
    fourth_error_test = zeros(params.class_Num+1,params.class_Num+1);
    for j = 1:params.class_Num
        for k = 1:params.info_Num
            dist = [];
            for jj = 1:params.class_Num
                for kk = 1:params.info_Num
                    dist(jj,kk) = norm(data(j).sample(i).info(k).cen_mon_norm-...
                                    data(jj).sample(1).info(kk).cen_mon_norm,2);
                end
            end
```

```
            [value,indice] = sort(dist(:),'ascend');
            [index, datasample] = find(dist <= value(5));
            index = min(index);
            %index = index(randi(length(index)));
            fourth_error_test(j,index) = fourth_error_test(j,index) + 1;
        end
    end

    sum_type = sum(fourth_error_test(1:params.class_Num,1:params.class_Num));
    for p = 1:params.class_Num
        fourth_error_test(p,params.class_Num+1) =
            params.info_Num-fourth_error_test(p,p);
        fourth_error_test(params.class_Num+1,p) = sum_type(p) -
            fourth_error_test(p,p);
    end

    sum_total = sum(fourth_error_test');
    fourth_error_test(params.class_Num+1,params.class_Num+1) =
        sum_total(params.class_Num+1);

    fourth_error = fourth_error + fourth_error_test;
    fprintf('%d\t',sum_total(params.class_Num+1));
end

fprintf('\n\n\n\n             ******************  Method 4: 5NN in moment space
    ******************\n');
print(fourth_error)

clearvars -except data params
```

Listing 8: deal_data_pixel.m

```
for i = 1: params.class_Num
    for j = 1:params.sample_Num
        for k = 1:params.info_Num
            data(i).sample(j).info(k).pixel =
                reshape(data(i).sample(j).info(k).image,[1,256]);
        end
    end
end

clearvars -except data params
```

Listing 9: classify_fifth_pixel.m

```
global sample_Set;
global class_Set;
```

```
third_error = zeros(params.class_Num+1,params.class_Num+1);

for i = 1:params.sample_Num
    third_error_test = zeros(params.class_Num+1,params.class_Num+1);
    for j = 1:params.class_Num
        for k = 1:params.info_Num
            dist = [];
            for jj = 1:params.class_Num
                for kk = 1:params.info_Num
                    dist(jj,kk) = norm(data(j).sample(i).info(k).pixel-...
                                    data(jj).sample(1).info(kk).pixel,2);
                end
            end
            min_dist = min(min(dist));
            [index, datasample] = find(dist == min_dist);
            index = min(index);
            third_error_test(j,index) = third_error_test(j,index) + 1;
        end
    end

    sum_type = sum(third_error_test(1:params.class_Num,1:params.class_Num));
    for p = 1:params.class_Num
        third_error_test(p,params.class_Num+1) =
            params.info_Num-third_error_test(p,p);
        third_error_test(params.class_Num+1,p) = sum_type(p) -
            third_error_test(p,p);
    end

    sum_total = sum(third_error_test');
    third_error_test(params.class_Num+1,params.class_Num+1) =
        sum_total(params.class_Num+1);

    third_error = third_error + third_error_test;
    fprintf('%d\t',sum_total(params.class_Num+1));
end

fprintf(' \n\n\n\n            ******************  Method 5: 1NN in pixel space
    ******************\n');
print(third_error)

clearvars -except data params
```

Listing 10: classify_sixth_pixel.m

```
global sample_Set;
global class_Set;

svmStruct = {};
```

```
for i = 1:params.class_Num
    for j = i+1:params.class_Num
        trainingData = [];
        group = [];
        for k = 1:params.info_Num
            trainingData = [trainingData;data(i).sample(1).info(k).pixel];
            group = [group;i];
            trainingData = [trainingData;data(j).sample(1).info(k).pixel];
            group = [group;j];
        end
        svmStruct{i,j} = svmtrain(trainingData,group);
    end
end

SVM_error = zeros(params.class_Num+1,params.class_Num+1);

for i = 1:params.sample_Num
       SVM_error_test = zeros(params.class_Num+1,params.class_Num+1);
    for j = 1:params.class_Num
        for k = 1:params.info_Num
            testResult = [];
            for ii = 1:params.class_Num
                for jj = ii+1:params.class_Num
                    testResult =
                        [testResult,svmclassify(svmStruct{ii,jj},data(j).sample(i).info(k).pixel)
                end
            end
            index = mode(testResult);
            SVM_error_test(j,index) = SVM_error_test(j,index) + 1;
        end
    end

    sum_type = sum(SVM_error_test(1:params.class_Num,1:params.class_Num));
    for p = 1:params.class_Num
        SVM_error_test(p,params.class_Num+1) =
            params.info_Num-SVM_error_test(p,p);
        SVM_error_test(params.class_Num+1,p) = sum_type(p) - SVM_error_test(p,p);
    end

    sum_total = sum(SVM_error_test');
    SVM_error_test(params.class_Num+1,params.class_Num+1) =
        sum_total(params.class_Num+1);

    SVM_error = SVM_error + SVM_error_test;
    fprintf('%d\t',sum_total(params.class_Num+1));
end
```

```
 fprintf('\n\n\n\n              ******************  Method 6: SVM in pixel space
    ******************\n');
print(SVM_error)

clearvars -except data params
```

Listing 11: cen_mon.m

```
function cen_mom = cen_mon(p,q,x_c,y_c,mat,params)

    N = params.image_Size;
    cen_mom = 0;
    for i = 1:N
        for j = 1:N
            cen_mom = cen_mom+(i-x_c)^p*(j-y_c)^q*mat(i,j);
        end
    end
end
```

Listing 12: discrim.m

```
function g = discrim(x,p,q)
    g = 0;

    for i = 1 : 256
        g = g + x(i) * log(p(i)/q(i)) + (1 - x(i)) * log((1-p(i))/(1-q(i)));
    end
end
```

Listing 13: print.m

```
function pr = print(M)

    global class_Set;
    fprintf('Classified as:\tB\tC\tD\tE\tI\tJ\tO\tR\tU\tV\tErrorTypeI\n')
    fprintf('True class\n')
    for i = 1:10
        fprintf('\t%s\t',class_Set(i));
        for j = 1:11
            if M(i,j) ~= 0
                fprintf('%d\t',M(i,j))
            else
                fprintf('\t')
            end
        end
        fprintf('\n')
    end

    fprintf('ErrorTypeII\t')
    for j = 1:10
        if M(11,j) ~= 0
            fprintf('%d\t',M(11,j))
        else
            fprintf('\t')
```

```
        end
    end
    fprintf('%d(Total Error)\n\n\n\n',M(11,11))
end
```