Part I: A General Framework for Solutions to Q1 – Q3

Basically, Q1 – Q3 are problems of investigating the performance of different techniques using synthetic datasets. In this part, I propose a general frame work for solving such problems. The results and conclusions for Q1-Q3 will be presented in Part II – Part IV and the matlab code is in Part V. Further, Minitab 14 is used for Design of Experiment (DOE) Analysis.

As aforementioned, our objective is to investigate the performance of different techniques. To be specific, we want to evaluate the performance of different methods (effects on classification error rate). Besides a restrained conclusion of which method is better on the specific datasets under investigation, we may interested in asking questions like:

1) Do the datasets used represent a comprehensive set of scenarios?
1) Is it possible to quantify the effects of different methods?
2) The conclusion is valid with what level of (statistical) confidence?

For this purpose, I propose to use the method of Design of Experiment (DOE) in finding answers to the above. To generate a set of datasets which represent possible scenarios comprehensively, we can specify a list of factors which are important characteristics of datasets and then use DOE to generate a design which specify the setting of datasets. For simplicity, I only consider 2 level factorial designs in this report. Table 1 list the factors and the corresponding levels considered in experiment.

Factors	Sample-to-variable ratio	Within-to-between scatter	Balance/Non-balanced class	Methods under investigation
Formula	$\frac{\#samples}{\#input\ variables}$	$\frac{\lVert\mu_1-\mu_2\rVert}{\sqrt{\lvert\Sigma_1\rVert\Sigma_2\rvert}}$	$\frac{\#\ samples\ in\ class1}{\#\ samples\ in\ class2}$	The methods under investigation
Low level	10	0.5	1	
High level	100	1	4	

*: For simplicity, we assume that $x \in R^2$; there are two classes $y \in \{-1,1\}$ and each class is normal with the corresponding mean μ_i and covariance matrix Σ_i, $i = 1,2$ .We further assume that $\Sigma_i's$ are diagonal, i.e. there are no correlation between individual x variables.

There are three factors each with 2 level, i.e. $2^3 = 8$ different settings. For each of the K methods under investigation, we generate L replicates for each of the 8 settings and collect the classification accuracy generated - $K*8*L = 8KL$ runs in total. Finally, we will construct a DOE model on classification accuracy. The DOE model will then quantify the effects of factors in determining the accuracy. The P values associated with each model coefficients represent the corresponding statistical confidence.

Part II – Solution to Q1

Below is a list of factors under investigation

Factors	S-V ratio	W-B scatter	n1-n2	Methods under investigation
Formula	$\frac{\#samples}{\#input\ variables}$	$\frac{\lVert\mu_1-\mu_2\rVert}{\sqrt{\lvert\Sigma_1\rvert\lvert\Sigma_2\rvert}}$	$\frac{\#samples\ in\ class1}{\#samples\ in\ class2}$	Fisher or the 2nd method
Low level	10	0.5	1	-1 (Fisher)
High level	100	1	4	1 (2nd method)

For this design, we first generate 3 replicates for each setting – 8*2*3=48 runs in total. We run Fisher 's method and the 2nd method in Q1 on each of the 48 datasets, record the training and testing percentage error obtained and then do factorial fit on the error rate with respect to the factors in the design. Below is the DOE model obtained on fitting training error:

Factorial Fit: tr versus S-V ratio, W-B scatter, n1-n2, Fisher/M

Estimated Effects and Coefficients for tr (coded units)

Term	Effect	Coef	SE Coef	T	P
Constant		14.226	0.6532	21.78	0.000
S-V ratio	2.363	1.181	0.6532	1.81	0.079
W-B scatter	-12.102	-6.051	0.6532	-9.26	0.000
n1-n2	0.702	0.351	0.6532	0.54	0.594
Fisher/M	3.560	1.780	0.6532	2.72	0.010
S-V ratio*W-B scatter	-2.785	-1.392	0.6532	-2.13	0.040
S-V ratio*n1-n2	1.280	0.640	0.6532	0.98	0.334
S-V ratio*Fisher/M	-1.278	-0.639	0.6532	-0.98	0.334
W-B scatter*n1-n2	-2.102	-1.051	0.6532	-1.61	0.116
W-B scatter*Fisher/M	0.207	0.104	0.6532	0.16	0.875
n1-n2*Fisher/M	-0.274	-0.137	0.6532	-0.21	0.835

S = 4.52584 R-Sq = 74.11% R-Sq(adj) = 67.11%

Analysis of Variance for tr (coded units)

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Main Effects	4	1982.49	1982.49	495.622	24.20	0.000
2-Way Interactions	6	186.73	186.73	31.122	1.52	0.199
Residual Error	37	757.88	757.88	20.483		
Lack of Fit	5	18.33	18.33	3.666	0.16	0.976
Pure Error	32	739.55	739.55	23.111		
Total	47	2927.10				

Interpretation of DOE results:

1) With 99% confidence we conclude that the effect of "Fisher/M" is significant – in other words, there is significant difference in training error between the method of fisher discriminant analysis and the 2nd method; further more, when all other factors are the same, the training error produced by fisher discriminant is on average 3.56% smaller than that of the 2nd method.

2) Further, we also have enough confidence (>95%) to conclude that: a) the factor of with-to-between scatter is very important. Specifically, when the ratio increases from 0.5 to 1, training error decreases by 12.102% on average and when the s-v ratio is on high-level, there will be another 2.785% decrease in training error.

Model on testing error

Factorial Fit: tst versus S-V ratio, B-W scatter, n1-n2, Fisher/M

Estimated Effects and Coefficients for tst (coded units)

Term	Effect	Coef	SE Coef	T	P
Constant		15.656	0.7132	21.95	0.000
S-V ratio	0.479	0.240	0.7132	0.34	0.739
B-W scatter	-12.146	-6.073	0.7132	-8.52	0.000
n1-n2	2.104	1.052	0.7132	1.48	0.149
Fisher/M	1.438	0.719	0.7132	1.01	0.320
S-V ratio*B-W scatter	-2.146	-1.073	0.7132	-1.50	0.141
S-V ratio*n1-n2	0.438	0.219	0.7132	0.31	0.761
S-V ratio*Fisher/M	-0.229	-0.115	0.7132	-0.16	0.873
B-W scatter*n1-n2	-2.771	-1.385	0.7132	-1.94	0.060
B-W scatter*Fisher/M	-0.021	-0.010	0.7132	-0.01	0.988
n1-n2*Fisher/M	-0.937	-0.469	0.7132	-0.66	0.515

S = 4.94094 R-Sq = 69.01% R-Sq(adj) = 60.64%

Analysis of Variance for tst (coded units)

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Main Effects	4	1850.94	1850.94	462.734	18.95	0.000
2-Way Interactions	6	160.86	160.86	26.811	1.10	0.382
Residual Error	37	903.28	903.28	24.413		
Lack of Fit	5	48.94	48.94	9.789	0.37	0.868
Pure Error	32	854.33	854.33	26.698		
Total	47	2915.08				

Interpretation:

1) Lack of Fit test indicates no evidence for model failure;
2) The model shows a general trend of an increased error rate of 1.438% by switching from Fisher's method to the 2nd method. However, P value shows that there is not enough evidence to support the above statement.

Recommendation:

To further investigate whether the factor of Fisher/M is significant in fitting testing error rate, we need to

1) add more replicates to the design for a better estimate of the standard deviation of coefficients, or
2) re-design the setting of datasets, e.g. consider other characteristics of datasets or choose other levels of factors in the design.

Part III – Solution to Q2

Below is a list of factors under investigation

Factors	S-V ratio	W-B scatter	n1-n2	Methods under investigation
Formula	$\frac{\#samples}{\#input\ variables}$	$\frac{\|\mu_1-\mu_2\|}{\sqrt{\|\Sigma_1\|\Sigma_2\|}}$	$\frac{\#\ samples\ in\ class1}{\#\ samples\ in\ class\ 2}$	Neural Network (NN) or SVM method
Low level	10	0.5	1	-1 (NN)
High level	100	1	4	1 (SVM)

For this design, we first generate 6 replicates for each setting – 8*2*6=96 runs in total. We run Fisher 's method and the 2nd method in Q1 on each of the 48 datasets, record the training and testing percentage error obtained and then do factorial fit on the error rate with respect to the factors in the design. Below is the DOE model obtained on fitting training error:

Factorial Fit: N/S_tr versus S-V ratio, W-B scatter, n1-n2, NN/SVM

Estimated Effects and Coefficients for N/S_tr (coded units)

Term	Effect	Coef	SE Coef	T	P
Constant		10.613	0.3843	27.62	0.000
S-V ratio	5.100	2.550	0.3843	6.64	0.000
W-B scatter	-11.975	-5.988	0.3843	-15.58	0.000
n1-n2	-4.100	-2.050	0.3843	-5.33	0.000
NN/SVM	3.079	1.540	0.3843	4.01	0.000
S-V ratio*W-B scatter	-1.017	-0.508	0.3843	-1.32	0.189
S-V ratio*n1-n2	-0.225	-0.113	0.3843	-0.29	0.770
S-V ratio*NN/SVM	-2.462	-1.231	0.3843	-3.20	0.002
W-B scatter*n1-n2	1.517	0.758	0.3843	1.97	0.052
W-B scatter*NN/SVM	-2.579	-1.290	0.3843	-3.36	0.001
n1-n2*NN/SVM	0.046	0.023	0.3843	0.06	0.953

S = 3.76519 R-Sq = 80.84% R-Sq(adj) = 78.58%

Analysis of Variance for N/S_tr (coded units)

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Main Effects	4	4696.8	4696.8	1174.21	82.83	0.000
2-Way Interactions	6	386.5	386.5	64.41	4.54	0.000
Residual Error	85	1205.0	1205.0	14.18		
Lack of Fit	5	157.7	157.7	31.54	2.41	0.044
Pure Error	80	1047.3	1047.3	13.09		
Total	95	6288.3				

Interpretation:

1) With large confidence (P=0.000) we conclude that when all other factors being equal, SVM on average, has a training error that is 3.079% higher than NN.
2) When sample-to-variable ratio is low (size of training set is small), SVM has training error 2.462% lower than NN (at 98% confidence level);
3) When within-to-between scatter is low (classes are close/overlapping), SVM has training error 2.579% lower than NN.

Part IV – Solution to Q3

Note that training error for nearest neighbor is zero by definition. We will focus only on comparing testing error rate for this set of methods. Further, we choose K=3 for K nearest neighbor method.

Since the design was setup in a 2 level factorial fashion, we need to do pairwise comparison between the three methods. Below are the DOE models for factorial fit on testing error.

1) Nearest Neighbor (NN) v.s. Parzen window

Factorial Fit: tst1 versus S-V ratio, W-B scatter, n1-n2, NN/Parzen

Estimated Effects and Coefficients for tst1 (coded units)

Term	Effect	Coef	SE Coef	T	P
Constant		19.750	0.8809	22.42	0.000
S-V ratio	-3.833	-1.917	0.8809	-2.18	0.036
W-B scatter	-18.417	-9.208	0.8809	-10.45	0.000
n1-n2	-1.792	-0.896	0.8809	-1.02	0.316
NN/Parzen	-2.042	-1.021	0.8809	-1.16	0.254
S-V ratio*W-B scatter	0.750	0.375	0.8809	0.43	0.673
S-V ratio*n1-n2	-1.792	-0.896	0.8809	-1.02	0.316
S-V ratio*NN/Parzen	-0.375	-0.187	0.8809	-0.21	0.833
W-B scatter*n1-n2	1.375	0.687	0.8809	0.78	0.440
W-B scatter*NN/Parzen	-0.542	-0.271	0.8809	-0.31	0.760
n1-n2*NN/Parzen	2.083	1.042	0.8809	1.18	0.245

S = 6.10337 R-Sq = 76.39% R-Sq(adj) = 70.01%

Analysis of Variance for tst1 (coded units)

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Main Effects	4	4334.96	4334.96	1083.74	29.09	0.000
2-Way Interactions	6	125.25	125.25	20.88	0.56	0.759
Residual Error	37	1378.29	1378.29	37.25		
Lack of Fit	5	80.96	80.96	16.19	0.40	0.846
Pure Error	32	1297.33	1297.33	40.54		
Total	47	5838.50				

Interpretation:

The model shows a general trend of decreased testing error rate by 2.042% by switching from Nearest Neighbor to Parzen window. However, we only have 74.6% confidence level for the above statement.

2) Nearest Neighbor (NN) v.s. K Nearest Neighbor (KNN)

Factorial Fit: tst2 versus S-V ratio, W-B scatter, n1-n2, NN/KNN

Estimated Effects and Coefficients for tst2 (coded units)

Term	Effect	Coef	SE Coef	T	P
Constant		19.760	0.9278	21.30	0.000
S-V ratio	-3.396	-1.698	0.9278	-1.83	0.075
W-B scatter	-16.812	-8.406	0.9278	-9.06	0.000
n1-n2	-2.854	-1.427	0.9278	-1.54	0.133
NN/KNN	-2.021	-1.010	0.9278	-1.09	0.283
S-V ratio*W-B scatter	-0.563	-0.281	0.9278	-0.30	0.763
S-V ratio*n1-n2	-3.271	-1.635	0.9278	-1.76	0.086
S-V ratio*NN/KNN	0.063	0.031	0.9278	0.03	0.973
W-B scatter*n1-n2	0.562	0.281	0.9278	0.30	0.763
W-B scatter*NN/KNN	1.062	0.531	0.9278	0.57	0.570
n1-n2*NN/KNN	1.021	0.510	0.9278	0.55	0.586

S = 6.42810 R-Sq = 71.52% R-Sq(adj) = 63.82%

Analysis of Variance for tst2 (coded units)

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Main Effects	4	3677.1	3677.1	919.27	22.25	0.000
2-Way Interactions	6	162.1	162.1	27.01	0.65	0.687
Residual Error	37	1528.9	1528.9	41.32		
Lack of Fit	5	114.7	114.7	22.94	0.52	0.760
Pure Error	32	1414.2	1414.2	44.19		
Total	47	5368.0				

Interpretation:

The model shows a general trend of decreased testing error of 2.021% by switching from NN to KNN. However, again, we only have 71.7% confidence level for the statement.

3) KNN v.s. Parzen

Factorial Fit: tst3 versus S-V ratio, W-B scatter, n1-n2, KNN/Parzen

Estimated Effects and Coefficients for tst3 (coded units)

Term	Effect	Coef	SE Coef	T	P
Constant		18.740	0.7796	24.04	0.000
S-V ratio	-3.771	-1.885	0.7796	-2.42	0.021
W-B scatter	-17.354	-8.677	0.7796	-11.13	0.000
n1-n2	-0.771	-0.385	0.7796	-0.49	0.624
KNN/Parzen	-0.021	-0.010	0.7796	-0.01	0.989
S-V ratio*W-B scatter	1.396	0.698	0.7796	0.90	0.376
S-V ratio*n1-n2	-2.854	-1.427	0.7796	-1.83	0.075
S-V ratio*KNN/Parzen	-0.437	-0.219	0.7796	-0.28	0.781
W-B scatter*n1-n2	-0.604	-0.302	0.7796	-0.39	0.701
W-B scatter*KNN/Parzen	-1.604	-0.802	0.7796	-1.03	0.310
n1-n2*KNN/Parzen	1.063	0.531	0.7796	0.68	0.500

S = 5.40141 R-Sq = 78.60% R-Sq(adj) = 72.81%

Analysis of Variance for tst3 (coded units)

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Main Effects	4	3791.8	3791.8	947.94	32.49	0.000
2-Way Interactions	6	172.2	172.2	28.71	0.98	0.450
Residual Error	37	1079.5	1079.5	29.18		
Lack of Fit	5	113.0	113.0	22.60	0.75	0.593
Pure Error	32	966.5	966.5	30.20		
Total	47	5043.5				

Interpretation:

Model shows the effect of factor “KNN/Parzen” is very small (0.021), and the associated p value is very large (98.9%). It is implied that at there is not much difference between the Parzen window method and the KNN.

Conclusion:

Basing on the experimental results, DOE analysis identified a statistical pattern of decreased accuracy of about 2% when switching from the method of NN to Parzen Window or KNN (with roughly 70% confidence level) and there is no significant difference between KNN and Parzen window in terms of classification error rate obtained.

Recommendation for further investigation:

DOE model identifies a general trend of decreased/increased error rate between methods by the sign of the effects. However, for many cases, there is not enough statistical evidence to support the statement. To further investigate, need to redesign the setting of datasets or to add more runs for a better estimate of the standard deviation of coefficients.

Part V - Matlab Code

Reference:

1) The toolbox of "SVMlight" is used to implement linear SVM classifier; More information about SVMlight can be found at http://www.cs.cornell.edu/people/tj/svm_light/

2) The toolbox of "Netlab' is used to implement the forword Neural Network More information about "Netlab" can be found at http://www.ncrg.aston.ac.uk/netlab/

1. M file for generating data

```
clc
clear
close all
%dataset setting

% % 1 ABC-(++-)
% xr1=100;xr2=100; % # of samples for each cluster
% mu1 = [2,1];mu2=[4,3.83];sigma1=[4,0;0,1];sigma2=[2,0;0,1.5];

% % 1 ABC-(+++)
% xr1=40;xr2=160; % # of samples for each cluster
% mu1 = [2,1];mu2=[4,3.83];sigma1=[4,0;0,1];sigma2=[2,0;0,1.5];

% % 1 ABC-(+--)
% xr1=100;xr2=100; % # of samples for each cluster
% mu1 = [2,1];mu2=[3,2.414];sigma1=[4,0;0,1];sigma2=[2,0;0,1.5];

% % 1 ABC-(+-+)
% xr1=40;xr2=160; % # of samples for each cluster
% mu1 = [2,1];mu2=[3,2.414];sigma1=[4,0;0,1];sigma2=[2,0;0,1.5];

% % 1 ABC-(-+-)
% xr1=10;xr2=10; % # of samples for each cluster
% mu1 = [2,1];mu2=[4,3.83];sigma1=[4,0;0,1];sigma2=[2,0;0,1.5];

% % % 1 ABC-(-++)
% xr1=4;xr2=16; % # of samples for each cluster
% mu1 = [2,1];mu2=[4,3.83];sigma1=[4,0;0,1];sigma2=[2,0;0,1.5];

% % 1 ABC-(---)
% xr1=10;xr2=10; % # of samples for each cluster
% mu1 = [2,1];mu2=[3,2.414];sigma1=[4,0;0,1];sigma2=[2,0;0,1.5];
```

```
% % 1 ABC-(--+)
xr1=4;xr2=16; % # of samples for each cluster
mu1 = [2,1];mu2=[3,2.414];sigma1=[4,0;0,1];sigma2=[2,0;0,1.5];

X1 = mvnrnd(mu1,sigma1,xr1);%generate samples of X using the 1st Gaussian
X2 = mvnrnd(mu2,sigma2,xr2);%generate samples of X using the 2nd Gaussian
X=[X1',X2']';
Y1=zeros(xr1,1);Y2=ones(xr2,1);;
Y=[Y1',Y2']';

figure,hold on;
plot(X1(:,1),X1(:,2),'r.');
plot(X2(:,1),X2(:,2),'b*');
hold off

% xr1=40;xr2=160; % # of samples for each cluster
% mu1 = [2,1];mu2=[5,5.243];sigma1=[4,0;0,1];sigma2=[2,0;0,1.5];
```

2. m file for implementing classifiers

```
clear
clc
close all

load GDOE_ABC_--+_6
trset=X;trlabel=Y+1;
load GDOE_ABC_--+_1

tst_set=X;tst_label=Y+1;

%fisher linear
trn.X=trset';trn.y=trlabel';
tst.X=tst_set';tst.y=tst_label';
model = fld(trn);
ypred = linclass(trn.X,model);
tr_error=cerror(ypred,trn.y);
ypred = linclass(tst.X,model);
tst_error=cerror(ypred,tst.y);
[tr_error,tst_error]

%2nd method
A1=find(trlabel==1);A2=find(trlabel==2);
mu1=mean(X(A1,:));mu2=mean(X(A2,:));

w=(mu1-mu2)/sqrt((mu1-mu2)*(mu1-mu2)');
T=X*w';T_tst=tst_set*w';
trn.X=T';trn.y=trlabel';
```

```
tst.X=T_tst';tst.y=tst_label';
model = fld(trn);
ypred = linclass(trn.X,model);
tr_error=cerror(ypred,trn.y);
ypred = linclass(tst.X,model);
tst_error=cerror(ypred,tst.y);
[tr_error,tst_error]

%Backpropagation Neural Network
net = mlp(2, 3, 1, 'linear');
[xr,xc]=size(trset);
Set up vector of options for the optimiser.
options = zeros(1,18);
options(1) = 0;                    %This provides display of error values.
options(9) = 0;                    %Check the gradient calculations.
options(14) = 100;          %Number of training cycles.

[net, options] = netopt(net, options, trset, trlabel, 'scg');
pred = mlpfwd(net, trset);
ypred=1*(pred<=1.5)+2*(pred>1.5);
tr_error=cerror(ypred,trn.y);
pred = mlpfwd(net, tst_set);
ypred=1*(pred<=1.5)+2*(pred>1.5);
tst_error=cerror(ypred,tst.y);
[tr_error,tst_error]

%Linear SVM
C=10;
Y_learn=2*(trlabel==2)-1;
net_1stage=svml(['model_1stage.txt'], 'Verbosity', 0,
 'Kernel',0,'KernelParam',[1 1 1],'C',C,'ComputeLOO',0);
[net_1stage, results]=svmltrain(net_1stage,trset,Y_learn);
out1=(svmlfwd(net_1stage,trset));
ypred=(out1>0)+1;
tr_error=cerror(ypred,trlabel);
out1=(svmlfwd(net_1stage,tst_set));
ypred=(out1>0)+1;
tst_error=cerror(ypred,tst_label);
[tr_error,tst_error]

%KNN K=3
model=knnrule(trn,3);
ypred=knnclass(trn.X,model);
tr_error=cerror(ypred,trn.y);
ypred=knnclass(tst.X,model);
tst_error=cerror(ypred,tst.y);
```

```
[tr_error,tst_error]

%KNN K=1
valY=[1,2]';
[ypred_tr,tabkppv,distance]=knn(trset,trlabel,valY,trset,1);
tr_error=cerror(ypred_tr,trn.y);
[ypred,tabkppv,distance]=knn(trset,trlabel,valY,tst_set,1);
tst_error=cerror(ypred,tst.y);
[tr_error,tst_error]

%Parzen Window
[xr,xc]=size(trset);
[xtr,xtc]=size(tst_set);

A1=find(trlabel==1);A2=find(trlabel==2);
for i=1:xr
   f_tr(i,1)= ksdensity(trset(A1,:),trset(i,:));
   f_tr(i,2)= ksdensity(trset(A2,:),trset(i,:));
end
[T,ypred]=max(f_tr');ypred=ypred';
tr_error=cerror(ypred,trlabel);
for i=1:xtr
   f_tst(i,1)= ksdensity(trset(A1,:),tst_set(i,:));
   f_tst(i,2)= ksdensity(trset(A2,:),tst_set(i,:));
end
[T,ypred]=max(f_tst');ypred=ypred';
tst_error=cerror(ypred,tst_label);
[tr_error,tst_error]
```