3. Part I

a. μ and Σ from the first 10 data samples:

$$\mu = \begin{bmatrix}0.8190 & -0.6271\end{bmatrix}^T, \quad \Sigma = \begin{bmatrix}0.7461 & -0.1474 \\ -0.1474 & 1.6047\end{bmatrix}$$

b. μ and Σ from the first 100 data samples:

$$\mu = \begin{bmatrix}0.9977 & -0.9725\end{bmatrix}^T, \quad \Sigma = \begin{bmatrix}2.2580 & 1.0856 \\ 1.0856 & 2.1439\end{bmatrix}$$

c. μ and Σ from the first 1000 data samples:

$$\mu = \begin{bmatrix}1.0222 & -0.9629\end{bmatrix}^T, \quad \Sigma = \begin{bmatrix}2.2118 & 1.1878 \\ 1.1878 & 2.0332\end{bmatrix}$$

c. μ and Σ from the first 10000 data samples:

$$\mu = \begin{bmatrix}0.9947 & -1.0027\end{bmatrix}^T, \quad \Sigma = \begin{bmatrix}1.9978 & 0.9643 \\ 0.9643 & 1.9237\end{bmatrix}$$

e. Parameter estimation errors

Measure 1:

case	a	b	c	d
ε	1.7935	0.3088	0.2883	0.0845

Measure 2:

case	a	b	c	d
ε_μ	0.2931	0.0195	0.0306	0.0042
ε_Σ	1.0075	0.1776	0.1646	0.0487

In *Measure 1*, the estimation error declines as more data are obtained, however it is not all the cases in *Measure 2* as we observe the estimation error in mean ε_μ and the estimation error in covariance ε_Σ differently. When 100 samples are used to learn parameter $\boldsymbol{\theta}$, the error in mean increase compared to when only 10 samples are being fed into the MLE estimator and this phenomenon can be explained by the sensitivity of the mean measure to outlier in the samples data.

e. Plot of first 100 data samples and 2D contours of estimated Gaussian pdf

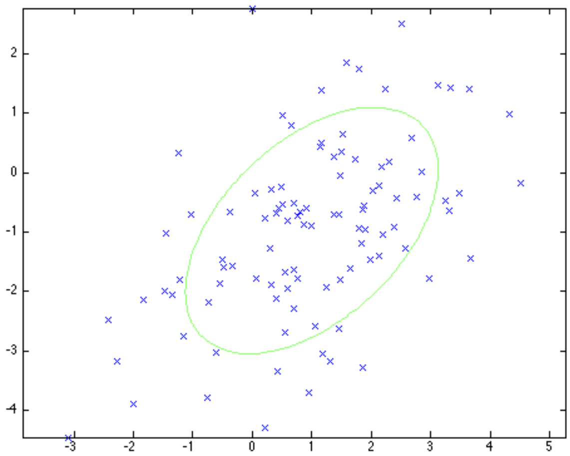

Figure 1: 100 data samples and estimated Gaussian pdf 2D contours

Part II

a. μ and Σ from the first 10 data samples:

$$\mu = \begin{bmatrix} 1.8829 & -1.8135 \end{bmatrix}^T, \quad \Sigma = \begin{bmatrix} 5.6385 & -5.3104 \\ -5.3104 & 5.3521 \end{bmatrix}$$

b. μ and Σ from the first 100 data samples:

$$\mu = \begin{bmatrix} 1.1741 & -1.2216 \end{bmatrix}^T, \quad \Sigma = \begin{bmatrix} 2.6753 & -2.5961 \\ -2.5961 & 2.6913 \end{bmatrix}$$

c. μ and Σ from the first 1000 data samples:

$$\mu = \begin{bmatrix} 0.9539 & -0.9530 \end{bmatrix}^T, \quad \Sigma = \begin{bmatrix} 1.9939 & -1.9344 \\ -1.9344 & 2.0528 \end{bmatrix}$$

c. μ and Σ from the first 10000 data samples:

$$\mu = \begin{bmatrix} 1.0023 & -1.0031 \end{bmatrix}^T, \quad \Sigma = \begin{bmatrix} 1.9659 & -1.8639 \\ -1.8639 & 1.9582 \end{bmatrix}$$

e. Parameter estimation errors

Measure 1:

case	a	b	c	d
ε	6.1275	1.2239	0.0914	0.0651

Measure 2:

case	a	b	c	d
ε_μ	0.8489	0.1993	0.0466	0.0028
ε_Σ	3.4692	0.6876	0.0366	0.0375

In both *Measure 1* and *Measure 2* we notice that parameter estimation errors decrease as the number of data samples increase. Maximum likelihood estimation assumes that the parameter $\boldsymbol{\theta}$ is fixed then seeks to find the parameter value that maximizes the probability of the training data being observed.

e. Plot of first 100 data samples and 2D contours of estimated Gaussian pdf

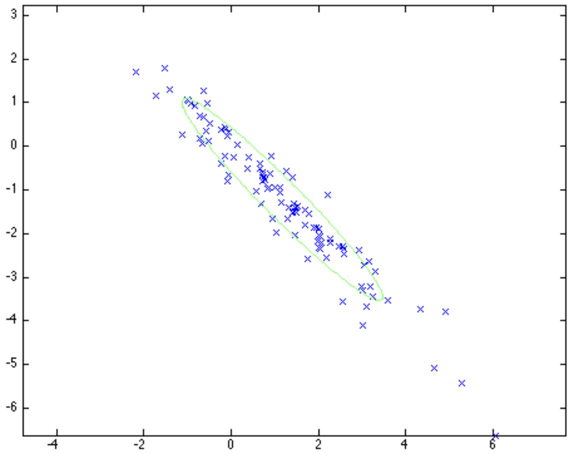

Figure 2: 100 data samples and estimated Gaussian pdf 2D contours

4.

a.

$$\mu_{MLE} = \begin{bmatrix} 4.2897 \\ 3.1275 \\ 2.9466 \\ 2.6994 \\ 2.5360 \\ 2.3195 \\ 2.1012 \\ 2.1773 \\ 2.1529 \\ 1.9803 \\ 1.9825 \\ 2.0185 \\ 1.9584 \\ 1.9170 \\ 1.9686 \\ 1.9283 \\ 1.9663 \\ 1.9756 \\ 1.9851 \\ 2.0310 \end{bmatrix}, \quad \mu_{MAP} = \begin{bmatrix} 2.6179 \\ 2.5975 \\ 2.6148 \\ 2.5178 \\ 2.4326 \\ 2.2876 \\ 2.1245 \\ 2.1821 \\ 2.1618 \\ 2.0185 \\ 2.0173 \\ 2.0454 \\ 1.9917 \\ 1.9535 \\ 1.9966 \\ 1.9593 \\ 1.9915 \\ 1.9986 \\ 2.0061 \\ 2.0467 \end{bmatrix}$$

b. Plot of the error curves of MLE and MAP:

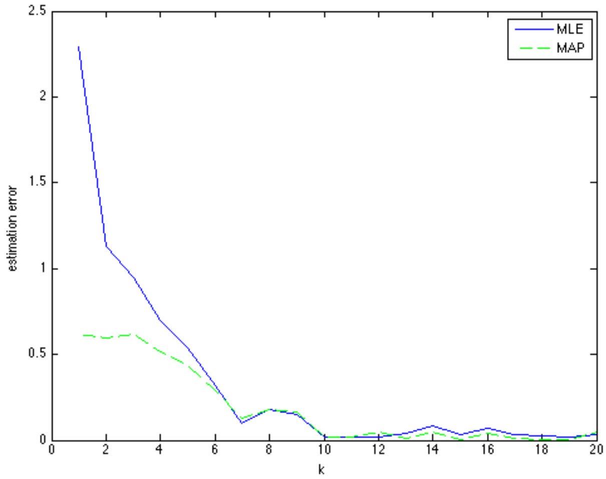

Figure 3: Error curves when $\mu_{true} = 2$

MLE performs poorly compared to MAP when there is less training data available and sometimes the error is too high to to be acceptable and should not be used however when samples data is abundant MLE estimation is almost as good as MAP method which requires much more computational power to evaluate.

c. Plot of the $\mu \sim N(\mu_N, \sigma_N^2)$ for k=1, 10 and 20:

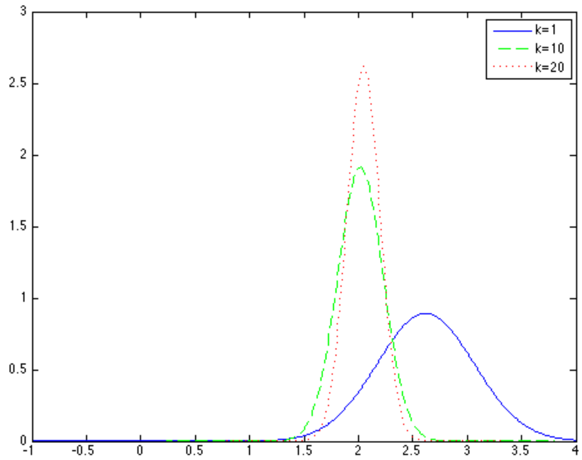

Figure 4: Error curves when $\mu_{true} = 2$

Appendix:

assignment_3.m

```
%
% CS7720 Spring 2015
% Introduction to Machine Learning and Pattern Recognition
% University of Missouri-Columbia
%
% Author: Chanmann Lim
% email: cl9p8@mail.missouri.edu
%
% Homework Assignment 3
% Problem 4
%
clc; clear; close all;

%%
% Problem 3. Part I
%
%   dataset - GDdataMLE1 dataset
%   m - true mean
%   P - true covariance

dataset = load('GDdataMLE1.txt');
m = [1; -1];
P = [2 1; 1 2];

problem_3_report;

%%
% Problem 3. Part II
%
%   dataset - GDdataMLE2 dataset
%   m - true mean
%   P - true covariance

dataset = load('GDdataMLE2.txt');
m = [1; -1];
P = [2 -1.9; -1.9 2];

problem_3_report;

%%
% Problem 4
%

dataset = load('GDdataMLEMAP.txt');
sigma = sqrt(2);
mu_0 = 2.2;
sigma_0 = sqrt(0.25);

problem_4_report;
```

problem_3_report.m

```
%
% Report for problem 3
%   m - mean
%   P - covariance

% a
[m_of_10_data_samples, P_of_10_data_samples] = mle(dataset(1:10, :));
display(m_of_10_data_samples);
display(P_of_10_data_samples);

% b
first_100_data_samples = dataset(1:100, :);
[m_of_100_data_samples, P_of_100_data_samples] = mle(first_100_data_samples);
display(m_of_100_data_samples);
display(P_of_100_data_samples);

% c
```

```
[m_of_1000_data_samples, P_of_1000_data_samples] = mle(dataset(1:1000, :));
display(m_of_1000_data_samples);
display(P_of_1000_data_samples);

% d
[m_of_10000_data_samples, P_of_10000_data_samples] = mle(dataset(1:10000, :));
display(m_of_10000_data_samples);
display(P_of_10000_data_samples);

% e
theta_true = theta(m, P);
theta_of_10_data_samples = theta(m_of_10_data_samples, P_of_10_data_samples);
theta_of_100_data_samples = theta(m_of_100_data_samples, P_of_100_data_samples);
theta_of_1000_data_samples = theta(m_of_1000_data_samples, P_of_1000_data_samples);
theta_of_10000_data_samples = theta(m_of_10000_data_samples, P_of_10000_data_samples);

error_1 = [
    error_measure_1(theta_of_10_data_samples, theta_true)
    error_measure_1(theta_of_100_data_samples, theta_true)
    error_measure_1(theta_of_1000_data_samples, theta_true)
    error_measure_1(theta_of_10000_data_samples, theta_true)
];
display(error_1);

error_2 = [
    error_measure_2(theta_of_10_data_samples, theta_true)
    error_measure_2(theta_of_100_data_samples, theta_true)
    error_measure_2(theta_of_1000_data_samples, theta_true)
    error_measure_2(theta_of_10000_data_samples, theta_true)
];
display(error_2);

% f
x1 = first_100_data_samples(:,1);
x2 = first_100_data_samples(:,2);

[X,Y]=meshgrid(-4:0.1:4, -4:0.1:4);
Pdf = normal2(X, Y, m_of_100_data_samples, P_of_100_data_samples);
levels = exp(-1) / ( 2*pi*sqrt( det(P_of_100_data_samples) ) );

figure;
plot(x1, x2, 'x'); hold on;
contour(X, Y, Pdf, [levels]);
axis equal;
```

mle.m

```
function [ m, P ] = mle( dataset )
% mle - Maximum likelihood estimator for mean and covariance
%       of 1-D and 2-D Gaussian dataset
%
%   m : the estimated mean (sample mean)
%   P : the estimated biased variance for 1-D dataset
%         and covariance matrix for 2-D dataset
%
% Note:
%   P = [var1 cov(1,2); cov(1,2) var2]
%
% where
%   var1        - biased variance of x1
%   cov(1, 2)   - E[(x1-mean_x1)(x2-mean_x2)]
%   var2        - biased variance of x2
%
    m = mean(dataset)';
    P = cov(dataset, 1);
end
```

theta.m

```
function [ theta ] = theta( m, P )
% theta - Construct theta vector given mean 'm' and covariance 'P'
%
% Output:
```

```
%    theta = [m1 m2 var1 cov(1,2) var2]'

    theta = [m; P([1 2 4])'];
end
```

error_measure_1.m

```
function [ e ] = error_measure_1( estimation, truth )
% error_measure_1 - Compute parameter estimation error
%    Where the error is L2-norm of the distance
%    between the estimation and the truth.
%
%    e = ||estimation - truth||
%
    distance = estimation - truth;
    e = sqrt( distance'*distance );
end
```

error_measure_2.m

```
function [ e ] = error_measure_2( estimation, truth )
% error_measure_2 - Compute parameter estimation error
%    error = [error_in_mean error_in_covariance]
%
% Where:
%    error_in_mean = ||estimation_mean - truth_mean|| / sqrt(2)
%    error_in_covariance = ||estimation_cov - truth_cov|| / sqrt(3)
%
    m_distance = estimation(1:2) - truth(1:2);
    P_distance = estimation(3:5) - truth(3:5);

    e = [
        sqrt( m_distance'*m_distance ) / sqrt(2) ...
        sqrt( P_distance'*P_distance ) / sqrt(3)
    ];
end
```