Math 128b Project

Jude Yuen

1. Introduction

Let $\{Z_t\}$ be a sequence of observed independent vector variables. If the elements of Z_t have a joint normal distribution, then $\{Z_t\}$ has a mean vector $\overline{Z}$ and a variance-covariance matrix Ω_z. Geometrically this is equivalent to that the Z_t's are clustering around the coordinate $\overline{Z}$ and is denser when it is closer to $\overline{Z}$. The changing of density around $\overline{Z}$ is governed by the variance-covariance matrix Ω_z. To estimate $\overline{Z}$ and Ω_z, a common practice in Statistics/Econometrics, is simply to maximize the likelihood function,

$$Max_q F(S;q) = \prod_{t=1}^{T} f\ (Z_t;\Omega_z,\overline{Z}), \tag{1}$$

Where,

$f\ (Z_t;\Omega_z,\overline{Z}) = (2p)^{-n/2} \mid \Omega_z \mid^{-1/2} \exp[-0.5(Z_t-\overline{Z})'\Omega_z^{-1}(Z_t-\overline{Z})]$,

$Z_1, Z_2, ..., Z_T \in S$,

$\overline{Z}, \Omega_z \in q$, and

n: the dimensions of Z_t.

Now suppose $\{y_t\}$ is a mixture of two sequences $\{Z_{t1}\}$ and $\{Z_{t2}\}$ each has different mean vector (m_1, m_2) and possibly different variance-covariance matrix (Ω_1, Ω_2) respectively. That is, some of the y_t's are clustering around m_1, and the rest around m_2, and the changing of density around m_1 and m_2 could be different because Ω_1 and Ω_2 may not be the same. If we can separate $\{y_t\}$ into $\{Z_{t1}\}$ and $\{Z_{t2}\}$, then (m_1, Ω_1) and (m_2, Ω_2) can be estimated separately as in (1). The challenge comes when we try to estimate (m_1, Ω_1) and (m_2, Ω_2), but *a priori,* we do not know which y_t belongs to which cluster. Then we need to find some ways to separate $\{Z_{t1}\}$ and $\{Z_{t2}\}$.

Studying clustering data is an important field in many disciplines. The mean vector and variance-covariance matrix of a given observed data set convey a lot of information about the observed data. And this information could be invaluable to business and economic study. For example, if Z_t is a 2-vector, one element is annual income and the other is annual consumption of an individual, and if $\{Z_t\}$ is clustering in 2 different places, knowing the mean vector and the variance-covariance matrix of each cluster, economists may be able to better predict what will happen and explain what is happening in an economy. And business executives may be able to better target individuals in each cluster with their products.

The most common method used in estimating the mean vectors and the variance-covariance matrices of normally distributed clustering data is the mixture likelihood model method. Prof. W. Kahan also suggests a way (the separate method) to separate the clusters and evaluate the mean vector and variance-covariance matrix of each cluster. This paper proposes a simple algorithm to estimate the mean vectors and the variance-covariance matrices of a data set with 2 normally distributed clusters. In comparing these methods, the proposed method seems to be more reliable and its speed is comparable to that of the separation method. The mixture likelihood method is the slowest, but it's fairly accurate.

This paper is organized in the following way. Next section will summarize the existing methods in estimating the mean vectors and the variance-covariance matrices of some normally distributed data consisting of 2 clusters. The proposed method will be outlined in section 3. Section 4 will compare the estimated results from the separation method, the proposed method, and the mixture likelihood model method for some simulated data. Section 5 concludes.

2. The Existing Methods

The common method used in estimating (m_1, Ω_1) and (m_2, Ω_2) in Statistics/Econometrics when $\{Z_{t1}\}$ and $\{Z_{t2}\}$ are both normally distributed is the mixture likelihood model approach (McLachlan and Basford, 1988). The idea behind the mixture likelihood model approach is that a fixed percentage (w_j) of the y_t's belong to cluster j. The mixture likelihood model approach will maximize the likelihood function:

$$Max_q \prod_{j=1}^{2} \prod_{i=1}^{T_j} p_{ji} f(y_i; \Omega_j, m_j) \qquad (2)$$

Where,

$$f(y_i; \Omega_j, m_j) = (2p)^{-n/2} |\Omega_j|^{-1/2} \exp[-0.5(y_i - m_j)' \Omega_j^{-1} (y_i - m_j)],$$

$$p_{ji}, \Omega_j, m_j \in q,$$

$$p_{ji} = \frac{w_j |\Omega_j|^{-1/2} \exp[-0.5(y_i - m_j)' \Omega_j^{-1} (y_i - m_j)]}{\sum_{j=1}^{2} w_j |\Omega_j|^{-1/2} \exp[-0.5(y_i - m_j)' \Omega_j^{-1} (y_i - m_j)]}, \qquad \text{the posterior Bayesian} \qquad (3)$$

probability that y_i belongs to cluster j.

$$w_j = \sum_{i=1}^{T} p_{ji} / T, \qquad \text{the percentage of data belongs to cluster j.} \qquad (4)$$

n: the dimensions of Z_t.

j=1,2. i =1,….,Tj

The algorithm to carry out the maximization of (2):

I. Pick the initial values for w_j^0, m_j^0, and Ω_j^0 for j=1,2.

II. Substitute the initial values into (3) to get the p_{ji}^0's.

III. Substitute the p_{ji}^0's into (2) to get the new estimates of m_j^1 and Ω_j^1, and into (4) to get the new estimates of w_j^1, for j=1,2.

IV. Stop if p_{ji}^k, m_j^k, and Ω_j^k converge; otherwise repeat step (II).

This algorithm is tantamount to weigh each y_i to see how much of it belongs to each cluster. If y_i has more weight toward cluster j, then y_i will contribute more to the calculation of the m_j^k and Ω_j^k in the iteration. This is also called the EM (Expectation & Maximization) algorithm. Expectation: p_{ji}^{k+1} the posterior Bayesian probability that y_i belongs to cluster j given w_j^k, m_j^k, and Ω_j^k. Maximization: maximizing (2) given p_{ji}^k.

Another closely related method which is often used in time series Econometrics to estimate the interested statistics [(m_1, Ω_1) and (m_2, Ω_2) in our case] is the Markov Regime Switching model method (Hamilton, 1989). The idea behind the Markov Regime Switching model method is that there are n (n=2, in our case) states of the world. When a particular state i of the world is realized in a given period, there is a fixed probability q_{ij} that the j state of the world will be realized in the next period for i, j = 1,2,….,n, $0 \le q_{ij} \le 1$, and $\sum_{j=1}^{n} q_{ij} = 1$. The Markov Regime Switching model method is basically the same as the mixture likelihood model method, but the Markov Regime Switching model method also imposes the assumption that a first order Markov process governs the probability that a particular vector in the sequence belongs to a particular cluster, a state of the world.

The numerical method for the Markov Regime Switching model is costly, complicated, and convergence is not guaranteed even for nicely distributed well-separated cluster data. Since the method is quite complicated, we will not get into the details of the Markov Regime Switching model here. Another drawback of the Markov Regime Switching model is that the model assumes the changing of the state of the world follows a Markov process. If the realization of a particular state of the world does not follow a Markov process, then the imposition of the Markov process will constrain the data to fit the Markov process and other statistical estimates. That, in turn, may bias all the statistical estimates. The numerical method for the mixture likelihood model is less complicated and costly, but convergence is also not guaranteed.

The separation method is quite straightforward. Given two clusters of data in a R^n Euclidean space, the separation method will find a hyperplane to separate the data into two clusters, and the hyperplane will pass the point (region) where the density is the lowest in between the clusters. To locate the point (region) with the lowest density, the norm vector of the hyperplane is first found by minimize and then maximize the following,

$$Max_n Min_c \frac{n'(Y - cu)n}{n'n}$$

Where

n: the norm of the hyperplane,

$Y := [y_1, y_2, ..., y_T]$

$u := [1,1,.....,1]$.

c turns out to be the mean vector of the whole data set, the center gravity of all the points in the data set. After finding the norm vector (n) of the hyperplane, place the hyperplane at (c), the center gravity of the all the points in the data set to separate the data into two sets temporarily. Then calculate the mean vectors of each set. Finally, the point (region) with the lowest density can be found along the norm in between the two mean vectors just found. Then the hyperplane can be placed there to separate two clusters. Obviously, convergence is not an issue here because no iteration is need. But if the data overlap too much, there may not be any least dense point along the norm vector (n) in between the two mean vectors just described

3. The Proposed Method

Just like the approach in estimating the mean vector and variance-covariance matrix in the single cluster vector sequence $\{Z_t\}$, we estimate (m_1, Ω_1) and (m_2, Ω_2) by maximizing the likelihood function, but now with respect to all four parameters,

$$Max_q F(S;q) = \prod_{t=1}^{T_1} f\ (y_t; \Omega_1, m_1) \prod_{t=1}^{T_2} f(y_t; \Omega_2, m_2) \qquad (5)$$

Where

$f\ (y_t;\Omega_j, m_j) = (2p)^{-n/2} \mid \Omega_j \mid^{-1/2} \exp[-0.5(y_t - m_j)'\Omega_j^{-1}(y_t - m_j)]$,

$y_1, y_2, ..., y_T \in S$,

$m_j, \Omega_j \in q$,

j =1,2, And

n: the dimensions of y_t.

Since $\{Z_{t1}\}$ and $\{Z_{t2}\}$ are independent, maximizing (3) is equivalent to maximizing $\prod_{t=1}^{T_1} f\ (y_t;\Omega_1, m_1)$ and $\prod_{t=1}^{T_2} f(y_t;\Omega_2, m_2)$ with respect to (m_1, Ω_1) and (m_2, Ω_2) in separation as in (1). The idea of the method proposed here is the same as in (1), the trick is to separate $\{y_t\}$ into $\{Z_{t1}\}$ and $\{Z_{t2}\}$.

The first order conditions,

$$\frac{\partial F(S;q)}{\partial m_j} = \Omega_j^{-1} \sum_{t=1}^{T_j} (y_t - m_j) = 0$$

$$=> \sum_{t=1}^{T_j} (y_t - m_j) = 0$$

$$=> \sum_{t=1}^{T_j} y_t = T_j * m_j$$

$$=> m_j = \frac{1}{T_j} \sum_{t=1}^{T_j} y_t$$

$$\frac{\partial F(S;q)}{\partial \Omega_j} = -0.5 * \frac{2p \mid \Omega_j \mid * \Omega_j^{-1}}{2p \mid \Omega_j \mid} + 0.5 * \Omega_j^{-2} \sum_{t=1}^{T_j} (y_t - m_j) * (y_t - m_j)' = 0$$

$$=>\Omega_j = \frac{1}{T_j}\sum_{t=1}^{T_j}(y_t - m_j)*(y_t - m_j)'$$

j = 1, 2,

and $T_1 + T_2 = T$.

Proposition: If m_1^*, m_2^*, Ω_1^*, and Ω_2^* satisfy the above first order conditions. Then $F(S;q^*) = \prod_{t=1}^{T_1} f\ (y_t;\Omega_1^*, m_1^*)\prod_{t=1}^{T_2} f(y_t;\Omega_2^*, m_2^*)$ is the global maximum.

Proof:

Since we know that if a set of parameters, q^*, satisfies the first order conditions in a maximization problem with a pseudo-concave objective function over a convex region, then the objective function attains its global maximum at q^*, to show the first order conditions are the sufficient conditions in the maximization problem above, we only need to show: i) the objective function is pseudo-concave, and ii) the feasible parameters set(s) are convex.

We first show the objective function is pseudo-concave. Since $f\ (y_t;\Omega_j, m_j) = (2p)^{-n/2} \mid \Omega_j \mid^{-1/2} \exp[-0.5(y_t - m_j)'\Omega_j^{-1}(y_t - m_j)]$ is the bell-shaped normal distribution density curve, it is pseudo-concave. And the objective function is the product of the pseudo-concave functions, so it is pseudo-concave. Now we show the feasible parameters set(s) is convex. Since there is no restriction on the parameter m_j and Ω_j, except that m_j is an n by 1 vector and Ω_j is an n by n symmetric positive definite matrix, $m_j \in R^2$ and R^2 is convex, and Ω_j belongs to the set of all n by n symmetric positive definite matrices which is also convex, the feasible region is convex. This completes the proof.

These first order conditions are quite conventional, and they just have been shown to be the necessary and sufficient conditions in maximizing the likelihood function $F(S;q) = \prod_{t=1}^{T_1} f\ (y_t;\Omega_1,m_1)\prod_{t=1}^{T_2} f(y_t;\Omega_2,m_2)$. Now the problem is to decide which y_t belongs to which cluster. The following algorithm should do the trick to separate the y_t's

The algorithm,

I. Pick the initial m_1^0, Ω_1^0 and m_2^0, Ω_2^0 values.

II. Substitute m_1^0, Ω_1^0 and then m_2^0, Ω_2^0 into the following equation to get two sequences of $f^{\,j}(y_t) = f(y_t;\Omega_j^k,m_j^k)$ (t = 1, 2, ……, T; and j = 1, 2) values,

$$f^{\,j}(y_t) = f(y_t;\Omega_j^k,m_j^k) = (2p)^{-n/2}\,|\Omega_j^k|^{-1/2}\ \exp[-0.5(y_t - m_j^k)'(\Omega_j^k)^{-1}(y_t - m_j^k)]$$

III. Compare the $f^{\,i}(y_t)$ values for each t, if $f^{\,1}(y_t)$ is greater than $f^{\,2}(y_t)$ assign y_t to set S_1^k. If $f^{\,2}(y_t)$ is greater than $f^{\,1}(y_t)$, assign y_t to set S_2^k. If $f^{\,1}(y_t)$ equals $f^{\,2}(y_t)$, then assign y_t to set S_3^k.

IV. Calculate the ratio:

$$r_j^k = \frac{cardinality(S_j^k)}{\sum_{j=1}^{2} cardinality(S_j^k)} \qquad j=1,2.$$

And assign $ceiling[(r_1^k) * cardinality(S_3^k)]$ numbers of y_t in set S_3^k to set S_1^k randomly, the rest to set S_2^k.

V. Calculate the mean vector and variance-covariance matrix of set S_j^k as m_j^{k+1} and Ω_j^{k+1}.

VI. If m_1^k, Ω_1^k, m_2^k, and Ω_2^k converge, then stop. Otherwise repeat step (II).

Troubled Initial Values

If some extreme initial values are picked such that the values from $f(y_t;\Omega_1^0,m_1^0)$ completely dominate the values from $f(y_t;\Omega_2^0,m_2^0)$ or vice verse, then obviously, the algorithm cannot go on. This problem can be dealt with easily by choosing the initial mean vectors and variance-covariance matrices near the mean vector and variance-covariance matrices of the whole data set. Still if m_j, Ω_j are picked such that,

$$\Omega_j = \frac{a(y_i - m_j)(y_i - m_j)'}{(y_i - m_j)'(y_i - m_j)} + e(I - \frac{(y_i - m_j)(y_i - m_j)'}{(y_i - m_j)'(y_i - m_j)}), \tag{8}$$

$a >> 0$ and $e \to 0$ for any given y_i, then $f(y_i;\Omega_j,m_j)$ will tend to infinity, and the algorithm will not work. Theoretically, this could happen. Since $f(y;\Omega,m)$ is a bell-shaped Gaussian curve and $\oint_{R^n} f(y;\Omega,m)dy = 1$, if the base of the bell is made very narrow, the peak of the bell could go to infinity. So if one or a few of the $f(y_t;\Omega,m)$ values approach infinity, and the rest are not zero, the likelihood function approaches infinity. Certainly, an initial point such as the one in (8) can be picked and obtain a very high or an infinite likelihood value. If the y_t's are not nicely distributed (that is one or a few y_t's are far away from the rest) the estimates generated by the algorithm could also converge to such problem spots.

Another possibility is that the m_1^0, Ω_1^0 and m_2^0, Ω_2^0 are picked such that exactly halves of the $\{Z_{t1}\}$ and $\{Z_{t2}\}$ in $\{y_t\}$ attain higher vales with m_1^0, Ω_1^0 and the other halves with m_2^0, Ω_2^0, then the algorithm may be stuck at the same point. We think such a point is not stable and the algorithm cannot converge to such a point. But this conjecture still needs to be proven.

Proposition: If $\{y_t\}$ consists of 2 nice, normally-distributed clusters (that is, no outliners are located far away from both clusters), the two clusters are well-separated, and the troubled initial values mentioned above are avoided, then the m_1^k, Ω_1^k, m_2^k, and Ω_2^k generated by the above algorithm will converge to m_1, Ω_1, m_2, and Ω_2.

Proof.

From the first order conditions, it is clear that to maximize the likelihood function (5), we simply have to pick the parameters (m_j, Ω_j) to generate as many high $f(y_t; \Omega_j, m_j)$ values as possible with $\{y_t\}$, j = 1, 2. Given a set of initial values (m_j^0, Ω_j^0), j = 1,2, without lost of generality, assume $f(y_t; \Omega_1, m_1)$ attains a higher value with more of the y_t's clustering around m_1 (for convenient, call this cluster A, and the other cluster B). Then m_1^1 is the mean of more y_t's in cluster A than those in cluster B. And Ω_1^1 is the average outer products of the y_t's mostly near cluster A minus m_1^1. Since there are only two clusters in the data, if the y_t's are not used in generating the (m_1^k, Ω_1^k), they are used in generating (m_2^k, Ω_2^k), what happens in cluster A is just the mirror image of what is happening in cluster B. Since m_1^1 and Ω_1^1 are generated by the y_t's mostly in cluster A, (m_1^1, Ω_1^1) are closer to (m_1, Ω_1) than (m_2^1, Ω_2^1) respectively. Similarly, (m_2^1, Ω_2^1) are closer to (m_2, Ω_2) than (m_1^1, Ω_1^1) respectively. Since $f(y_t; \Omega_j, m_j)$ is a bell-shaped Gaussian curve and $\oint_{R^n} f(y; \Omega_j, m_j)dy = 1$, $f(y_t; \Omega_j, m_j)$ is small if y_t is far away from m_j. That is for a given (m^*, Ω^*), $f(y_t; \Omega^*, m^*)$ attains a higher value if y_t is closer to m^*. Since m_1^1 is closer to m_1 than m_2^1, it is closer to most of the y_t's in cluster A than those from cluster B, and the same is true for m_2^1 with respect to cluster B. So more y_t's in cluster A will be included in calculating (m_1^2, Ω_1^2). Successively, (m_1^k, Ω_1^k) will get closer and closer to (m_1, Ω_1), and (m_2^k, Ω_2^k), to (m_2, Ω_2). As (m_1^k, Ω_1^k) get close enough to (m_1, Ω_1), $f(y_t; \Omega_1^k, m_1^k)$ will attain a higher value than

$f(y_t;\Omega_2^k,m_2^k)$ with every y_t's in cluster A. The same is true for $f(y_t;\Omega_2^k,m_2^k)$ with every y_t's in cluster B. Consequently, m_1^k, Ω_1^k, m_2^k, and Ω_2^k will converge to m_1, Ω_1, m_2, and Ω_2 respectively.

4. Methods Comparison

In this section we will compare the separation method, the proposed method, and the mixture likelihood method in terms of the numbers of iterations and computing time (since no iteration is need for the separation method, only the computing time (as in MatLab) of the method is compared to the other two methods) needed for the estimates to converge, and the sum of the norms of the differences between the true mean vectors, variance-covariance matrices and the respective estimated mean vectors, variance-covariance matrices.

The method of generating the comparing figures: two vector sequences with different mean vectors and variance-covariance matrices are generated. The mean vector and variance-covariance matrix of each sequence are calculated as the true mean vector and variance-covariance matrix. Then the two sequences are mixed, the separation method, the proposed method, and the mixture likelihood method are employed to estimate the mean vector and the variance-covariance matrix of each sequence. The amount of time and the numbers of iterations needed for the estimates to converge for each method are recorded. Also the norms of the differences between the true mean vectors and the variance-covariance matrices and the mean vectors and variance-covariance matrices estimated by each method are calculated.

Since the data generated or observed are random, if the cluster overlap too much, without knowing beforehand, there is no way one can tell which data point belongs to which cluster in the overlapping area. So we make an arbitrary decision that if the clusters overlap too much, roughly more than ten percent, we will advise the user of the proposed algorithm to look at the estimates with caution. The percentage of overlapping is

estimated first by calculating the distance between the estimated mean vectors of the two clusters A and B. Then draw a hyperplane (P_B), normal to the vector which runs from the mean vector of cluster A to that of cluster B, to pass through the mean vector of cluster B. Then calculate the average distance (D_B) of all the points in cluster B which lie on the side of the hyperplane (P_B) away from cluster A. Follow the same procedure to calculate D_A. If y_t is 3 dimensional, when the distance between the estimated mean vectors is less than $1.2*(D_A + D_B)$, the clusters should overlap each other more than ten percent. Then we send out the warning message to urge the user of this method to look at the estimates with caution. Under this circumstance we estimate the variance-covariance matrix of cluster B based on the data that lie on the side of P_B that is away from cluster A in each iteration. The same method is used to estimate the variance-covariance matrix of cluster A.

Now we move on to look at the experiment results. We first look at how does each method do with well-separated clusters as in figure 1 in the Appendix. Data sets with 2 well-separated clusters and a total of 1000 and 10000 data points were generated. And 1000 experiments for each scenario were repeated, but only the results of the last ten experiments and the summaries figures are reported in Table 1 and Table 2 in the Appendix.

In the table, under the heading of each method, the first column, *Time* is the amount of time needed for the estimates generated by the method to converge, for the separation method, it is the time need to generate the estimates. *Iterations* is the numbers of iterations needed for the estimates generated by the method to converge. *Norms* is the sum of the norms as described in the previous paragraph. *N.C.* is the time when the estimates generated by the method do not converge in that experiment (after 2000 iterations), then the column takes the value of one. Since *Iterations* and *N.C.* do not apply to the separation method, we just put N/A in these columns for the separation method. The first ten rows of the table are the results of last ten of the one thousand experiments. The next to the last row is the sum of each column and the last row is the average of each column.

As we can see from Table 1 and Table 2 in the Appendix, the message is consistent from both tables. The proposed method is the fastest and has the smallest Norms. The separation method comes in second. The mixture likelihood method is the slowest and has the biggest norms. The numbers of iterations for the proposed method and the mixture likelihood method are consistent with the computing time: the proposed method is about three times faster than the mixture likelihood method. In one occasion under each scenario, the estimates from mixture likelihood method did not converge.

For the clusters that are closed but may or may not overlap as in Figure 2 in the Appendix, the experiment with 1000 and 10000 data points are reported in Table 3 and Table 4. Table 3 is very much like Table 1 and Table 2. From Table 4, when the clusters are closed, the separation method becomes the fastest, but the norms is also the biggest, more than twice as big as the proposed method. The mixture likelihood method is still about three times as costly as the proposed method in terms of computing time or numbers of iterations. Also about two percent of the time, the estimates for the mixture likelihood method did not converge.

Lastly, we look at the case where the clusters overlap as in Figure 3 in the Appendix. The results for 1000 and 10000 data points are reported in Table 5 and Table 6. The results in Table 5 and Table 6 are similar to those in Table 4. The separation method is the fastest, but the proposed method has the smallest norms. In the case of 1000 data points, about one percent of the time, the estimates from the proposed method did not converge. And about two and a half percent of the time the estimates from mixture likelihood method did not converge. In Table 6 (the 10000 data points case) about five percent of the time, estimates from the mixture likelihood method did not converge. This figure seems a bit too high. But since the data were random, mostly likely the mixture likelihood method was just "lucky" to get such a high figure.

5. Conclusion

In this paper a simple algorithm is proposed to estimate the mean vectors and the variance-covariance matrices of two clusters of data, which are not overlapping too much. The results produced by the proposed method are also compared to those produced by the separation method and the mixture likelihood method for some simulated data under different scenarios. Clearly, when the clusters are well-separated, the proposed method stands out in terms of speed and accuracy. When the clusters are closed or overlap, the separation method is the fastest, but the proposed method is the most accurate. The mixture likelihood method is fairly accurate but it is relatively slow.

Reference

Hamilton, James, 1990, “Analysis of Time Series Subject to Changes in Regime,” *Journal of Econometrics* 45, 39-70.

Mclachlan, Geoffrey J., Kaye E. Basford, 1988, *Mixture Models: Inference and Applications to Clustering*, New York: Marcel Dekker.

Appendix

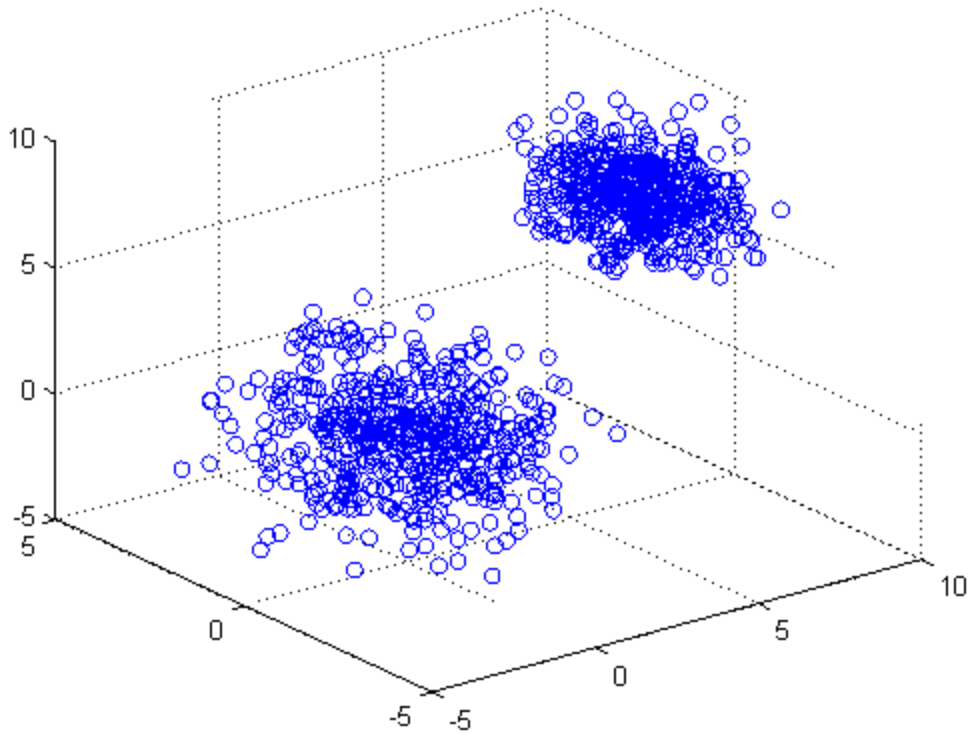

Figure 1

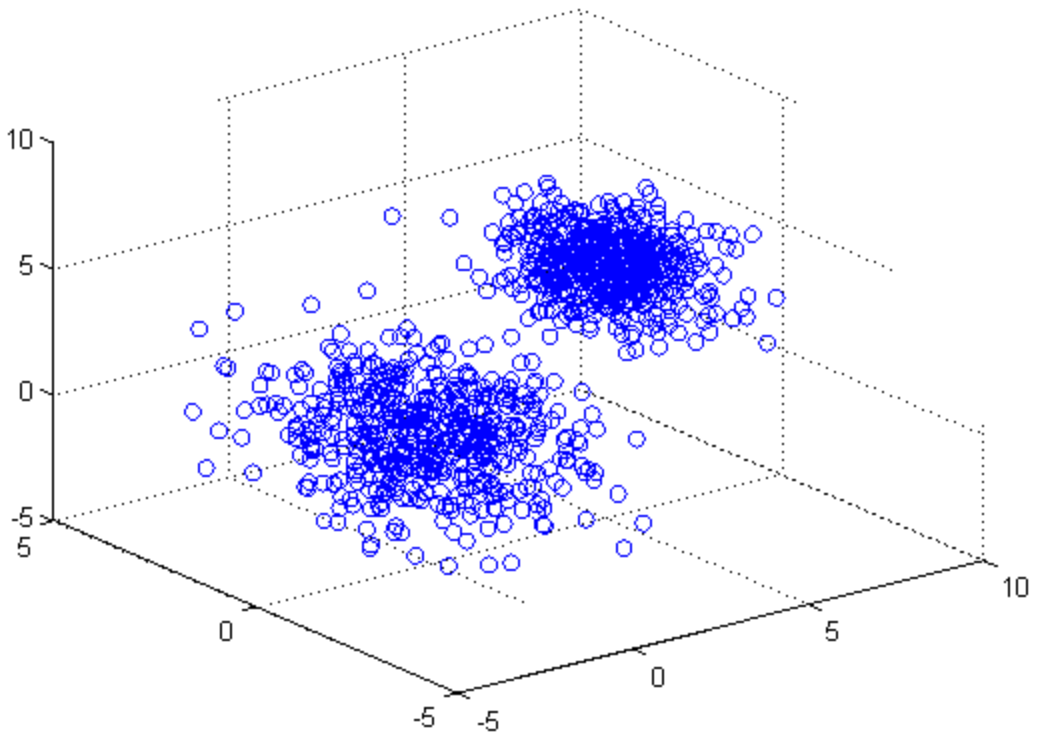

Figure 2

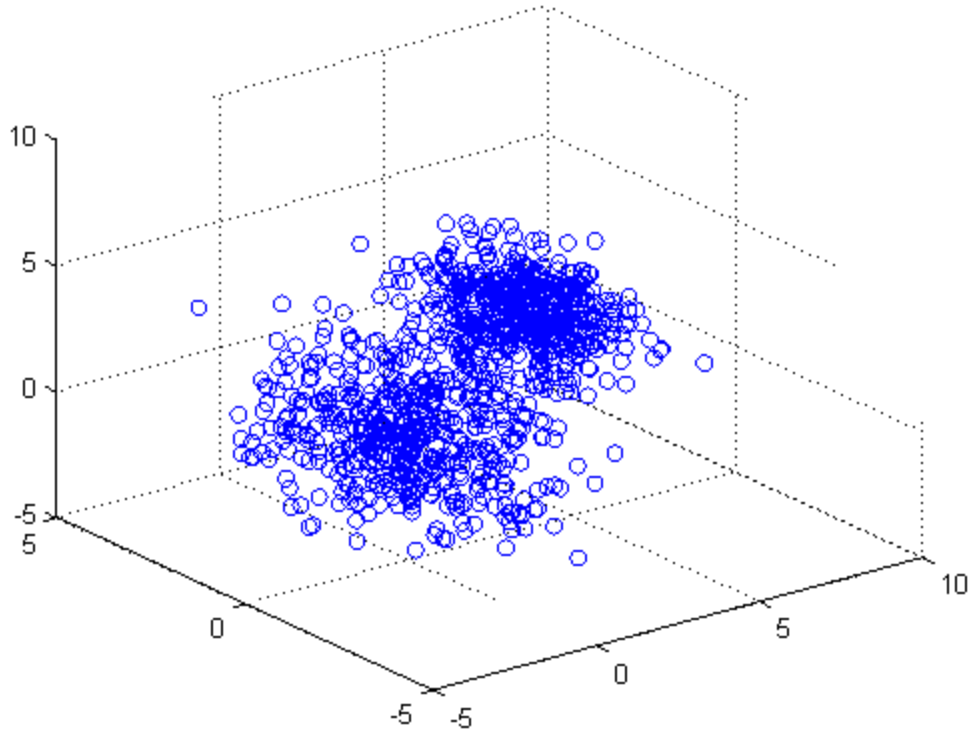

Figure 3

Table 1 With 1000 Well-Separated Data Point											
Separation Method				Proposed Method				Mixture Likelihood			
Time	Iterations	Norms	N.C.	Time	Iterations	Norms	N.C.	Time	Iterations	Norms	N.C.
0.0200	0.0000	0.0034	N/A	0.0100	3.0000	0.0034	0.0000	0.0400	11.0000	0.0069	0.0000
0.0100	0.0000	0.0035	N/A	0.0100	2.0000	0.0035	0.0000	0.0810	21.0000	0.0507	0.0000
0.0200	0.0000	0.0037	N/A	0.0200	3.0000	0.0037	0.0000	0.0300	7.0000	0.0038	0.0000
0.0200	0.0000	0.0036	N/A	0.0100	2.0000	0.0036	0.0000	0.0300	9.0000	0.0047	0.0000
0.0200	0.0000	0.0035	N/A	0.0200	3.0000	0.0035	0.0000	0.0300	9.0000	0.0046	0.0000
0.0100	0.0000	0.0035	N/A	0.0100	3.0000	0.0035	0.0000	0.0300	7.0000	0.0036	0.0000
0.0200	0.0000	0.0037	N/A	0.0100	3.0000	0.0037	0.0000	0.0300	7.0000	0.0037	0.0000
0.0200	0.0000	0.0038	N/A	0.0100	2.0000	0.0038	0.0000	0.0300	7.0000	0.0040	0.0000
0.0200	0.0000	0.0036	N/A	0.0200	3.0000	0.0036	0.0000	0.0300	9.0000	0.0045	0.0000
0.0200	0.0000	0.0036	N/A	0.0100	2.0000	0.0036	0.0000	0.0400	10.0000	0.0061	0.0000
18.2050	0.0000	10.8985	N/A	10.8330	2556.0000	7.6304	0.0000	31.6990	8547.0000	9.0057	1.0000
0.0182	0.0000	0.0109	N/A	0.0108	2.5560	0.0076	0.0000	0.0317	8.5470	0.0090	0.0010

Note: ***Time*** is the amount of time needed for convergence in MatLab. ***Iterations*** is the numbers of iterations need for convergence. ***Norms*** is the sum of norms of the differences of the true statistics and the respective estimated statistics**.** ***N.C.*** equals 1 if the program does not converge after 2000 iterations. ***N/A***: not applied. The first ten rows are the last ten results of the one thousand experiments. The next to last row is the sum of each column, the last row is the average.

Table 2 With 10000 Well-Separated Data Points											
Separation Method				Proposed Method				Mixture Likelihood			
Time	Iterations	Norms	N.C.	Time	Iterations	Norms	N.C.	Time	Iterations	Norms	N.C.
0.1500	N/A	0.0003	N/A	0.1300	3.0000	0.0003	0.0000	0.3710	10.0000	0.0061	0.0000
0.1500	N/A	0.0003	N/A	0.1300	3.0000	0.0003	0.0000	0.4110	11.0000	0.0031	0.0000
0.1500	N/A	0.0003	N/A	0.1300	3.0000	0.0003	0.0000	0.3010	8.0000	0.0009	0.0000
0.1500	N/A	0.0157	N/A	0.1300	3.0000	0.0003	0.0000	0.3710	10.0000	0.0041	0.0000
0.1500	N/A	0.0003	N/A	0.1300	3.0000	0.0003	0.0000	0.3010	8.0000	0.0010	0.0000
0.1500	N/A	0.0132	N/A	0.1400	3.0000	0.0003	0.0000	0.4010	11.0000	0.0060	0.0000
0.1500	N/A	0.0003	N/A	0.1900	4.0000	0.0138	0.0000	0.4110	11.0000	0.0073	0.0000
0.1500	N/A	0.0158	N/A	0.1400	3.0000	0.0003	0.0000	0.4110	11.0000	0.0060	0.0000
0.1500	N/A	0.0003	N/A	0.1300	3.0000	0.0003	0.0000	0.3810	10.0000	0.0030	0.0000
0.1500	N/A	0.0119	N/A	0.1300	3.0000	0.0119	0.0000	0.4210	11.0000	0.0074	0.0000
150.0290	N/A	6.1664	N/A	137.9580	3051.0000	3.9086	0.0000	361.6930	9624.0000	4.9602	1.0000
0.1500	N/A	0.0062	N/A	0.1380	3.0510	0.0039	0.0000	0.3617	9.6240	0.0050	0.0010

Note: ***Time*** is the amount of time needed for convergence in MatLab. ***Iterations*** is the numbers of iterations need for convergence. ***Norms*** is the sum of norms of the differences of the true statistics and the respective estimated statistics**.** ***N.C.*** equals 1 if the program does not converge after 2000 iterations. ***N/A***: not applied. The first ten rows are the last ten results of the one thousand experiments. The next to last row is the sum of each column, the last row is the average.

Table 3 With 1000 Closed Data Points											
Separation Method				Proposed Method				Mixture Likelihood			
Time	Iterations	Norms	N.C.	Time	Iterations	Norms	N.C.	Time	Iterations	Norms	N.C.
0.0100	0.0000	0.1358	N/A	0.0200	4.0000	0.1358	0.0000	0.0810	21.0000	0.1603	0.0000
0.0200	0.0000	0.2759	N/A	0.0200	3.0000	0.1343	0.0000	0.0900	25.0000	0.1593	0.0000
0.0200	0.0000	0.9335	N/A	0.0200	4.0000	0.3890	0.0000	0.0000	0.0000	0.0000	1.0000
0.0100	0.0000	0.2962	N/A	0.0200	3.0000	0.1317	0.0000	0.1100	31.0000	0.1285	0.0000
0.0200	0.0000	0.0035	N/A	0.0200	3.0000	0.0035	0.0000	0.0800	23.0000	0.0624	0.0000
0.0300	0.0000	0.3019	N/A	0.0100	3.0000	0.0879	0.0000	0.1010	26.0000	0.1409	0.0000
0.0200	0.0000	0.0900	N/A	0.0200	4.0000	0.0900	0.0000	0.0900	27.0000	0.1401	0.0000
0.0100	0.0000	0.3992	N/A	0.0300	4.0000	0.0875	0.0000	0.1000	28.0000	0.1348	0.0000
0.0200	0.0000	0.3136	N/A	0.0200	4.0000	0.1936	0.0000	0.1000	27.0000	0.1778	0.0000
0.0200	0.0000	0.0644	N/A	0.0100	3.0000	0.0644	0.0000	0.0800	20.0000	0.0924	0.0000
18.6480	0.0000	177.1844	N/A	17.1350	3590.0000	105.1713	0.0000	85.5960	23035.0000	121.8572	22.0000
0.0186	0.0000	0.1772	N/A	0.0171	3.5900	0.1052	0.0000	0.0856	23.0350	0.1219	0.0220

Note: ***Time*** is the amount of time needed for convergence in MatLab. ***Iterations*** is the numbers of iterations need for convergence. ***Norms*** is the sum of norms of the differences of the true statistics and the respective estimated statistics**.** ***N.C.*** equals 1 if the program does not converge after 2000 iterations. ***N/A***: not applied. The first ten rows are the last ten results of the one thousand experiments. The next to last row is the sum of each column, the last row is the average.

Table 4 With 10000 Closed Data Points											
Separation Method				Proposed Method				Mixture Likelihood			
Time	Iterations	Norms	N.C.	Time	Iterations	Norms	N.C.	Time	Iterations	Norms	N.C.
0.1510	N/A	0.0392	N/A	0.1900	4.0000	0.0373	0.0000	0.8410	23.0000	0.0906	0.0000
0.1500	N/A	0.1049	N/A	0.2410	5.0000	0.0456	0.0000	0.8110	22.0000	0.0830	0.0000
0.1500	N/A	0.0789	N/A	0.2300	5.0000	0.0485	0.0000	0.8120	22.0000	0.0890	0.0000
0.1500	N/A	0.1508	N/A	0.2400	5.0000	0.0658	0.0000	0.8910	24.0000	0.1051	0.0000
0.1510	N/A	0.0377	N/A	0.1900	4.0000	0.0555	0.0000	0.8210	22.0000	0.1020	0.0000
0.1500	N/A	0.2250	N/A	0.2410	5.0000	0.0933	0.0000	0.9310	25.0000	0.1237	0.0000
0.1500	N/A	0.0309	N/A	0.1910	4.0000	0.0339	0.0000	0.9310	25.0000	0.1015	0.0000
0.1500	N/A	0.0881	N/A	0.1900	4.0000	0.0395	0.0000	0.8220	22.0000	0.0833	0.0000
0.1500	N/A	0.0988	N/A	0.1900	4.0000	0.0873	0.0000	0.9010	24.0000	0.1048	0.0000
0.1510	N/A	0.0477	N/A	0.2400	5.0000	0.0276	0.0000	0.8210	22.0000	0.0804	0.0000
150.1380	N/A	111.3727	N/A	220.3830	4594.0000	50.8061	0.0000	835.8510	22396.0000	91.1561	24.0000
0.1501	N/A	0.1114	N/A	0.2204	4.5940	0.0508	0.0000	0.8359	22.3960	0.0912	0.0240

Note: ***Time*** is the amount of time needed for convergence in MatLab. ***Iterations*** is the numbers of iterations need for convergence. ***Norms*** is the sum of norms of the differences of the true statistics and the respective estimated statistics**.** ***N.C.*** equals 1 if the program does not converge after 2000 iterations. ***N/A***: not applied. The first ten rows are the last ten results of the one thousand experiments. The next to last row is the sum of each column, the last row is the average.

Table 5 With 1000 Overlapping Data Points											
Separation Method				Proposed Method				Mixture Likelihood			
Time	Iterations	Norms	N.C.	Time	Iterations	Norms	N.C.	Time	Iterations	Norms	N.C.
0.0200	0.0000	0.3177	N/A	0.0200	5.0000	0.6129	0.0000	0.2200	60.0000	0.5154	0.0000
0.0200	0.0000	0.4654	N/A	0.0300	6.0000	0.6661	0.0000	0.2710	75.0000	0.6097	0.0000
0.0200	0.0000	0.7195	N/A	0.0300	7.0000	0.4316	0.0000	0.2700	75.0000	0.4217	0.0000
0.0200	0.0000	1.8858	N/A	0.0200	5.0000	0.2596	0.0000	0.2910	81.0000	0.5203	0.0000
0.0100	0.0000	1.1770	N/A	0.0300	5.0000	0.4865	0.0000	0.2400	66.0000	0.5719	0.0000
0.0200	0.0000	0.6200	N/A	0.0200	5.0000	0.2407	0.0000	0.1800	50.0000	0.2871	0.0000
0.0210	0.0000	0.5516	N/A	0.0200	6.0000	0.5103	0.0000	0.2500	68.0000	0.5981	0.0000
0.0200	0.0000	0.2515	N/A	0.0300	7.0000	0.2566	0.0000	0.1800	51.0000	0.4151	0.0000
0.0200	0.0000	0.7115	N/A	0.0200	5.0000	0.4030	0.0000	0.3110	84.0000	0.5821	0.0000
0.0200	0.0000	0.2471	N/A	0.0300	5.0000	0.4971	0.0000	0.2300	65.0000	0.4645	0.0000
18.7520	0.0000	673.6066	N/A	27.3250	5729.0000	428.3384	11.0000	237.9720	66097.0000	522.6922	24.0000
0.0188	0.0000	0.6736	N/A	0.0273	5.7290	0.4283	0.0110	0.2380	66.0970	0.5227	0.0240

Note: ***Time*** is the amount of time needed for convergence in MatLab. ***Iterations*** is the numbers of iterations need for convergence. ***Norms*** is the sum of norms of the differences of the true statistics and the respective estimated statistics**.** ***N.C.*** equals 1 if the program does not converge after 2000 iterations. ***N/A***: not applied. The first ten rows are the last ten results of the one thousand experiments. The next to last row is the sum of each column, the last row is the average.

Table 6 With 10000 Overlapping Data Points											
Separation Method				Proposed Method				Mixture Likelihood			
Time	Iterations	Norms	N.C.	Time	Iterations	Norms	N.C.	Time	Iterations	Norms	N.C.
0.1500	N/A	0.2771	N/A	0.3900	8.0000	0.2487	0.0000	2.3040	63.0000	0.4258	0.0000
0.1600	N/A	0.4086	N/A	0.3400	7.0000	0.3451	0.0000	2.2140	60.0000	0.4770	0.0000
0.1500	N/A	0.5152	N/A	0.3500	7.0000	0.3119	0.0000	2.4340	65.0000	0.4752	0.0000
0.1600	N/A	0.4742	N/A	0.3510	7.0000	0.2745	0.0000	2.4930	67.0000	0.4338	0.0000
0.1500	N/A	0.5628	N/A	0.3510	7.0000	0.2789	0.0000	2.4130	65.0000	0.4123	0.0000
0.1610	N/A	0.6030	N/A	0.3500	7.0000	0.3013	0.0000	2.3140	62.0000	0.4576	0.0000
0.1500	N/A	0.4307	N/A	0.4000	8.0000	0.3438	0.0000	2.5040	67.0000	0.4543	0.0000
0.1600	N/A	0.7283	N/A	0.2910	6.0000	0.2875	0.0000	2.3430	63.0000	0.4335	0.0000
0.1500	N/A	0.7084	N/A	0.3510	7.0000	0.3261	0.0000	2.3130	62.0000	0.4618	0.0000
0.1500	N/A	0.4063	N/A	0.4010	8.0000	0.3214	0.0000	2.3530	63.0000	0.4534	0.0000
152.4720	N/A	545.5960	N/A	386.8970	7717.0000	331.6460	0.0000	2318.6200	62309.0000	445.0488	51.0000
0.1525	N/A	0.5456	N/A	0.3869	7.7170	0.3316	0.0000	2.3186	62.3090	0.4450	0.0510

Note: ***Time*** is the amount of time needed for convergence in MatLab. ***Iterations*** is the numbers of iterations need for convergence. ***Norms*** is the sum of norms of the differences of the true statistics and the respective estimated statistics**.** ***N.C.*** equals 1 if the program does not converge after 2000 iterations. ***N/A***: not applied. The first ten rows are the last ten results of the one thousand experiments. The next to last row is the sum of each column, the last row is the average.