Christopher Micek, Guanyi Mou, Bowen Tang, Hao Yin
Group 9
CS 534
4/1/2020

Assignment 2 Writeup

Part 1

Kalman Filter

A Kalman filter[1] is an estimator for Gaussian random variables that is robust to various types of noise and other inaccuracies. Given a model describing the evolution of the variable to be estimated and measurements of the variable at time points of interest (each subject to some amount of uncertainty), a Kalman filter combines these to form an estimate that quickly evolves to be more accurate than any one of its components. Here we describe the Kalman filter we have developed to estimate the month-by-month homeless population in Worcester. Our filter and the associated data are provided in the included Excel spreadsheet "Part 1.xlsx".

The Kalman filter assumes that the system state at time t, $\mathbf{x}_t$, is determined by the linear dynamical system

$$\mathbf{x}_t = \mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{B}_t\mathbf{u}_t + \mathbf{w}_t$$

where

- $\mathbf{x}_{t-1}$ is the previous state,
- $\mathbf{F}_t$ is the transition model,
- $\mathbf{u}_t$ is the control vector,
- $\mathbf{B}_t$ is the control-input model, and
- $\mathbf{w}_t$ is the process noise, drawn from a standard normal distribution with covariance matrix $\mathbf{Q}_t$ $\left(\mathbf{w}_t \sim \mathcal{N}(0, \mathbf{Q}_t)\right)$.

Here $\mathbf{x}_t$ is a one-element vector containing the true homeless population at time t. This equation contains our filter's transition model (how we expect the population to change over time), plus a term capturing whatever the uncertainty is in the distribution the true latent state is drawn from.

For the data from each of the $1 \leq i \leq n$ sensors, $\mathbf{z}_{i,t}$, measuring our state, we also have

$$\mathbf{z}_{i,t} = \mathbf{H}_{i,t}\mathbf{x}_t + \mathbf{v}_{i,t}$$

where

- $\mathbf{H}_{i,t}$ is the observation model (how the state space relates to the observed space), and

[1] Most of the description and equations here are paraphrased from Wikipedia.

- $\mathbf{v}_{i,t}$ is the observation noise, drawn from a standard normal distribution with covariance matrix $\mathbf{R}_{i,t}$ $\left(\mathbf{v}_{i,t} \sim \mathcal{N}\left(0, \mathbf{R}_{i,t}\right)\right)$.

We go into further detail about what we chose for our sensors and initial parameters in the **Parameter Selection** section below. Our filter implementation is presented in the "Filtering+Prediction" workbook of the Excel spreadsheet.

The Kalman filter operates in two distinct phases: a "Predict" phase using the transition model, and an "Update" phase incorporating data from the sensor model. Both phases typically operate in an alternating fashion to produce an estimate, but the "Update" phase can be omitted or repeated, as necessary, depending on how many sensor observations are available. When sensor data is available, our filter performs two updates, since we have data from two sensors. The equations for both phases are below:

Predict

First an estimate of the next state, $\hat{\mathbf{x}}_{t|t-1}$, is generated, given the updated estimate of the previous state $\left(\hat{\mathbf{x}}_{t-1|t-1}\right)$, and the transition model:

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{F}_t \hat{\mathbf{x}}_{t-1|t-1} + \mathbf{B}_t \mathbf{u}_t.$$

Then, the covariance matrix of this estimate, $\mathbf{P}_{t|t-1}$, is given by

$$\mathbf{P}_{t|t-1} = \mathbf{F}_t \mathbf{P}_{t-1|t-1} \mathbf{F}_t^{\mathrm{T}} + \mathbf{Q}_t.$$

Update

If we have sensor observations we can use to improve our estimate, we update our previous estimate by weighting the transition model with the sensor model to generate the new estimate, $\hat{\mathbf{x}}_{i,t|t}$, for sensor i at time t given all measurements up to and including t:

$$\hat{\mathbf{x}}_{i,t|t} = \left(\mathbf{I} - \mathbf{K}_{i,t} \mathbf{H}_{i,t}\right) \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_{i,t} \mathbf{z}_{i,t},$$

where $\mathbf{K}_{i,t}$ is the Kalman gain that weights the estimate from the transition model against the t^{th} observation from the i^{th} sensor. The Kalman gain is given by:

$$\mathbf{K}_{i,t} = \mathbf{P}_{t|t-1} \mathbf{H}_{i,t}^{\mathrm{T}} \left(\mathbf{H}_{i,t} \mathbf{P}_{t|t-1} \mathbf{H}_{i,t}^{\mathrm{T}} + \mathbf{R}_{i,t}\right)^{-1}.$$

The updated estimate covariance, $\mathbf{P}_{i,t|t}$ is

$$\mathbf{P}_{i,t|t} = \left(\mathbf{I} - \mathbf{K}_{i,t} \mathbf{H}_{i,t}\right) \mathbf{P}_{t|t-1}.$$

In the case of multiple sensors, the "Update" step is repeated for each subsequent available sensor, replacing the parameters from the prediction step with the previous outputs of the update equations above, e.g.

$$\mathbf{K}_{2,t} = \mathbf{P}_{1,t|t} \mathbf{H}_{2,t}^{\mathrm{T}} \left(\mathbf{H}_{2,t} \mathbf{P}_{1,t|t} \mathbf{H}_{2,t}^{\mathrm{T}} + \mathbf{R}_{2,t}\right)^{-1},$$

$$\hat{\mathbf{x}}_{2,t|t} = \left(\mathbf{I} - \mathbf{K}_{2,t} \mathbf{H}_{2,t}\right) \hat{\mathbf{x}}_{1,t|t} + \mathbf{K}_{2,t} \mathbf{z}_{2,t},$$

$$\mathbf{P}_{2,t|t} = \left(\mathbf{I} - \mathbf{K}_{2,t}\mathbf{H}_{2,t}\right)\mathbf{P}_{1,t|t}.$$

Parameter Selection

First, we note that if $\mathbf{x}_t$, $\mathbf{u}_t$, and $\mathbf{z}_{i,t}$ are all one-element vectors, all the matrices in the equations above are 1×1, and can stand in for single numbers. Second, we assume, aside from the state and sensor observation vectors and anything we calculate in the "Update" and "Prediction" steps, that all other parameters are not time varying. This assumption makes sense once we specify our transition model and sensors.

To inform our parameter selection, we used two datasets: the 2010–2018 population estimates for Worcester City[2] from the US Census Bureau and the 2006–2017 point-in-time Worcester City homeless population counts[3]. This data and the associated calculations are presented in the "Worcester City Population Data" workbook in "Part 1.xlsx." In the cases below, the goal was to come up with numbers that were plausible for our model parameters and make a working filter, so some of our calculations and statistics might not be 100% correct.

Our dynamical system model should reflect both how we think the homeless population should change over time (the transition model), as well as any uncertainty in how well the model fits the world (the process noise). To get an idea for how this looked in real data, we graphed the point-in-time counts of the homeless population in Worcester City from 2006 to 2017 and computed a linear regression to see if there was any systematic change over time, and found that the homeless count increases by ≈ 16.95 people/year according to the linear trend (so ≈ 1.38 people/month). This will be our $\mathbf{u}_t$, since it does not depend on the homeless population from the previous time point.

To determine the uncertainty in our transition model, $\mathbf{Q}_t = \sigma_x^2$, we detrended the homeless count data (so it was centered on zero) and calculated the sample variance of the result. This captures the spread of the year-to-year change in homeless population that falls outside of the linear fit, so it makes sense as a description of how badly the model performs (if the model was perfect, the detrended data should be very close to zero, and the variance would be small). The calculated variance for yearly homeless counts was $\approx 6{,}850.1$, so to scale it to our monthly data, we have

$$\mathbf{Q}_t = \sigma_x^2 \approx \frac{6{,}850.1}{12} \approx 570.84.$$

Therefore, our dynamical system is

$$\mathbf{x}_t \approx \mathbf{x}_{t-1} + 1.38 + \mathbf{w}_t,$$

where $\mathbf{F}_t = \mathbf{B}_t = 1$ from our previous equations, and $\mathbf{w}_t \sim \mathcal{N}(0, \mathbf{Q}_t = \sigma_x^2 \approx 570.84)$.

To determine the uncertainty in our initial estimate, $\mathbf{P}_{0|0} = \sigma_t^2$ (the population variance), we look at the table on page 135 in *Housing the Homeless* by Erickson & Wilhelm (1986), which

[2] Worcester City American Community Survey 5-year estimates, 2010–2018

[3] Central Massachusetts Housing Alliance Point-in-Time Homeless Counts

states that in an estimate of the homeless population in Worcester, MA in the early 1980s, the most reliable range was from 1,500–1,900 people. If we take this to be a 95% confidence interval, then $1{,}900 - 1{,}500 = 400 = 4\sigma_t$, and $\sigma_t = 100$. We are given that the homeless population in Worcester in March, 2020 is 1,200, so we scale the variance to correspond to this new distribution mean:

$$\sigma_t^2 = \left(100 * \frac{1{,}200}{\frac{1{,}500 + 1{,}900}{2}} \right)^2 \approx 311.42.$$

Now a large portion of our model has been defined: we have our linear dynamical system, which we use to generate the true state each month (shown in the cells labeled "true change" and "true effective mean"), and the transition model, which we use for our "Predict" phase (the cells labeled "predicted change", "predicted effective mean", and "effective variance"). We interpreted the supplied 1,200 population as being the true effective mean for March, which we index as $t + 1$. This is because our predicted change for March should come from the true value for *February*, the previous month, which we index as t. We reverse engineer the February's true population as follows:

$$\mathbf{x}_t \sim \mathcal{N}(1{,}200 - \mathbf{X}, \sigma_t^2 \approx 311.42),$$

where

$$\mathbf{X} \sim \mathcal{N}(\mathbf{u}_t \approx 1.38, \sigma_x^2 \approx 570.84)$$

and represents the population change from February to March.

Our filter uses data from two sensors: one based on a hypothetical point-in-time homeless count, and another based on total population estimates of Worcester City. Since the sensors are noisy measurements, taken from a normal distribution centered on the true value with some variance, defining the variance for each sensor is enough to define our sensor model.

For the first sensor, we take its variance to be the uncertainty we calculated previously for our initial estimate; that is, $\mathbf{R}_{1,t} = \sigma_{z,1}^2 = \sigma_t^2 \approx 311.42$. Since we will be taking point-in-time counts each month, this makes sense, as the initial estimate is derived from a point-in-time count, and there is no reason we would expect the uncertainty from the same type of measurement to change from month to month.

For our second sensor, we take more of an indirect approach: according to the National Alliance to End Homelessness[4], there are 22.6 homeless per 10,000 people in the general population. Using the census data estimates from [2], we calculated the expected number of homeless people in Worcester from 2010–2018, and used our previous method of finding the sample variance of the detrended data to reach a variance of ≈ 0.1019 when extrapolating from

[4] National Alliance to End Homelessness 2018 statistics

the full population. Additionally, the census data also includes margins of error in the form of 90% confidence intervals[5] for each year's estimates. For each year,

$$\text{MOE} = 1.645\sigma,$$

so

$$\sigma^2 = \left(\frac{\text{MOE}}{1.645}\right)^2.$$

The mean yearly variance for measuring the total population obtained from this method is $\approx 1{,}540.1$. Combining this with the previous result, and adjusting for monthly sampling, yields

$$\sigma_{z,2}^2 \approx \frac{1{,}540.1 + 0.1019}{12} \approx 128.35.$$

Our fully defined filter is simplified considerably from the general case. $\mathbf{H}_{i,t} = \text{I} = 1$ in our equations above, and the equation for the Kalman gain is

$$\mathbf{K}_{i,t} = \frac{\sigma_{i,t|t-1}^2}{\sigma_{i,t|t-1}^2 + \sigma_x^2}.$$

The filter is implemented with these parameters in the "Filtering+Prediction" workbook in "Part 1.xlsx." Monthly estimates after "Predict" and "Update" steps are in the cells labeled "new population 2." The workbook hews closely to the format used in class for the "kalman filter-1.xlsx" spreadsheet.

Smoothing

After generating monthly predictions, we are asked by our boss to use our estimates from the previous six months to make revised estimates. This is a case of smoothing, since we are looking to revise our estimate of past population including information we have now that was unavailable in the past.

We implemented Rauch-Tung-Striebel (RTS)[6] smoothing for our Kalman filter in the "Smoothing" workbook in "Part 1.xlsx." The smoothing works in two parts: forward calculations, which are taken directly from the unaltered Kalman filter, and backward calculations, which smooth previous estimates using future information, beginning at the last time point. The backward smoothing works in the exact opposite order from the forward estimation: while before we had homeless population estimates from each sensor in the order $\text{Sensor}_{1,\text{Mar}} \to \text{Sensor}_{2,\text{Mar}} \to \cdots \to \text{Sensor}_{1,\text{Aug}} \to \text{Sensor}_{2,\text{Aug}}$, the backward calculations revise the previous estimates in the order $\text{Sensor}_{2,\text{Aug}} \to \text{Sensor}_{1,\text{Aug}} \to \cdots \to \text{Sensor}_{2,\text{Mar}} \to \text{Sensor}_{1,\text{Aug}}$. First, we initialize the smoothed values for Sensor 2 in August to be the final state and variance estimates from the forward filtering, and then proceed backwards using a different set of equations. To avoid confusion here, we discretize time in the following equations in terms

[5] Using American Community Survey Estimates and Margins of Error

[6] https://cse.sc.edu/~terejanu/files/tutorialKS.pdf

of estimates from update steps from the forward process: t refers to the current update, and $t+1$ refers to the next update (from the opposite sensor).

The smoothing updates use a modified Kalman gain that makes use of the estimate from the next time point:

$$\mathbf{K}_t^S = \mathbf{P}_{t|t}\mathbf{F}_t\left(\mathbf{P}_{t+1|t}\right)^{-1} = \frac{\sigma_t^2}{\sigma_t^2 + \sigma_x^2}.$$

The smoothing update equations are:

$$\hat{\mathbf{x}}_{t|t}^S = \hat{\mathbf{x}}_{t|t} + \mathbf{K}_t^S\left(\hat{\mathbf{x}}_{t+1|t+1}^S - \hat{\mathbf{x}}_{t+1|t}\right) = \hat{\mathbf{x}}_{t|t} + \mathbf{K}_t^S\left(\hat{\mathbf{x}}_{t+1|t+1}^S - \hat{\mathbf{x}}_{t|t} - \mathbf{u}_t\right)$$

$$\mathbf{P}_{t|t}^S = \mathbf{P}_{t|t} + \mathbf{K}_t^S\left(\mathbf{P}_{t+1|t+1}^S - \mathbf{P}_{t+1|t}\right)(\mathbf{K}_t^S)^{\mathrm{T}} = \sigma_t^2 + (\mathbf{K}_t^S)^2\left(\mathbf{P}_{t+1|t+1}^S - \sigma_t^2 - \sigma_x^2\right)$$

The "Smoothing" workbook contains a plot comparing the filtered estimates, smoothed estimates, and "ground truth" data, as well as a plot showing the difference between smoothed and filtered estimates for each month. The result seems quite hit or miss, at least with this implementation; the smoothed estimate does look like a smoothed-out version of the original estimates, and is sometimes closer to the ground truth data than the filtered estimate, but sometimes is farther away as well. It is difficult to say definitively, but it appears that the smoothed estimate deviates most from the filtered estimate preceding or following a sharp change in the ground truth data. It is in situations like this that knowing future estimates matters; the smoother is better able to quickly "correct course" when faced with a sensor input that bucks the trend compared to the ordinary filter.

Prediction

For September and October, we are asked to predict the future homeless population without and sensor data. We are also given the information that the new mayor will adopt policies that will make the homeless population increase. Our chosen interpretation of this scenario is that the exact increase caused by the implementation of these policies is uncertain; these are uncharted seas, and such policies are without precedent in modern history. Therefore, rather than assume that we know exactly what the policy impact will be, we modify our process noise to be drawn from a half-normal distribution. That is, $\mathbf{w}_t \sim |\mathcal{N}(0, \sigma_x^2)|$, where $\mathbf{w}_t \geq 0$. Thus the homeless population is guaranteed to increase, but by an unknown amount. To attempt to account for this somehow, we also double our control vector in the transition model: $\mathbf{u}_t' = 2 \times \mathbf{u}_t \approx 2 \times 1.38 \approx 2.77$. This intuitively makes sense since now the population is effectively twice as likely to increase as before.

Since we have no sensor input, our predictions are entirely based on the transition model (this occurs from row 223 and beyond in the "Filtering+Prediction" workbook in "Part 1.xlsx"). Absent the sensor data the prediction performs terribly poorly in the face of overwhelming uncertainty (though it does capture the broad trend), and we are fired. ☹

Part 2

Expectation Maximization (EM)

We apply the EM algorithm in soft clustering problems, where we design a program which absorbs the multi-dimensional data instances and then outputs the corresponding clustering results. In soft clustering, each instance is assigned with confidence/probability of its belonging to a certain cluster, and each cluster is assumed to follow a Gaussian distribution. The task is to predict/learn the best parameters of each cluster's mean, covariance and weight. If a positive integer is given for the target number of clusters, the program will look for the provided number of clusters. If a zero is passed into the program, then the program will try to find out the best number of clusters guided by BIC.

1. **Cluster initialization**
    - **How:** We randomly pick a proportion of the data instances and calculate their mean, covariance and set up the weight to be the same. Here in the final version of the code, the ratio of instances is set to 0.3
    - **Was it random?** Yes, the instances are randomly selected, but to guarantee that our program is reproducible, we have set the random seed. One can easily change the manual fixed random seed to anything else. E.g. use time.time()
    - **Did we methodically walk through the search space?** We believe so. As the cluster centers are designed in the way described above, they cannot be too far away from the actual centers however still guaranteeing randomness.
    - **Any other thoughts?** We thought of other candidate mechanisms for initialization. For example, running a simple KNN on the data instances for a few steps and take these centers as the cluster centers. However, in order to take most use of the time (we were given at most 10 seconds for the experiment), we chose the current method.
2. **Bayesian information criterion** is a criterion for model selection. The model with lowest BIC is preferred. The equation is:

$$BIC = \ln(N)K - 2\ln(L).$$

Where

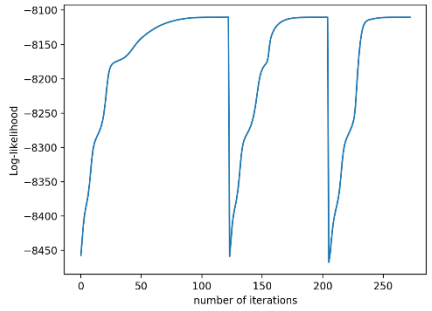
- K = the number of parameters estimated by the model. In EM clustering, K equals the number of clusters times the sum of the number of dimensions and the square of the number of dimensions.
- N = the number of data points.
- Ln(L) = Loglikelihood.

In our algorithm, when choosing 0 as the initial number of clusters, we will go through 0 to the number of data point as the number of clusters and calculate the best result, which has the highest loglikelihood, for each cluster. The terminate condition is time

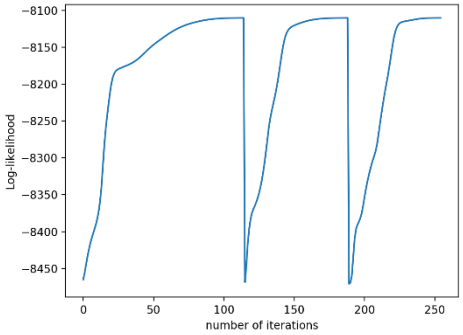
limits. When 10 seconds are reached, the search will stop and output the number of clusters with corresponding best likelihood. Then, we calculate the BIC for the best result from the different number of clusters and select the number of clusters with the lowest BIC.

3. **Hyper-Parameter Settings**
    - **Hypers:** We had explicitly set up hyper-parameters for number of iterations, random restarts and number of clusters.
    - **Tolerance/Convergence criteria:** Under the requirement/restriction of a time limit, we set up a tolerance threshold for determining convergence for each restart: once two consecutive iterations have a log-likelihood difference lower than the threshold, we stop the simulation in that epoch and restart. After trying several thresholds, we eventually chose its value to be 0.01.

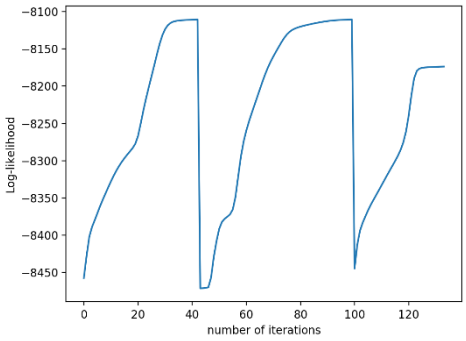
Tolerance: 0.001 loglikelihood: -8110.282301207892 time: 7 sec 760 msec



Tolerance: 0.01 Loglikelihood: -8110.316732582465 Time: 6 sec 968 msec

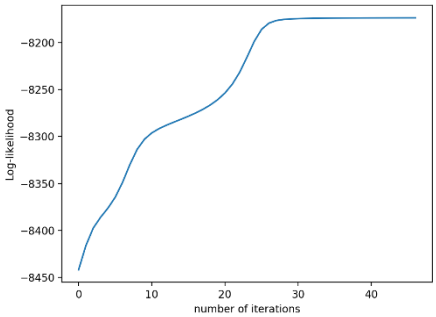


Tolerance: 0.1 Loglikelihood: -8173.774849038102 Time: 3 sec 458 msec

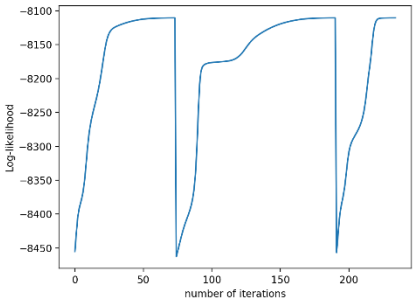


- **Random restarts:** the more #epochs/random restarts we have, the higher chance we arrive at global optimum. We tested several different numbers and eventually manually set it to be 3, to guarantee the improvement while not wasting too much time.

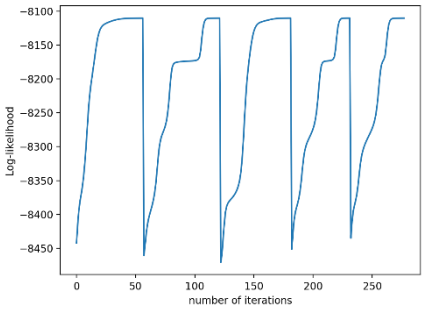
Restart: 1 Loglikelihood: -8173.441462870002 Time: 1 sec 320 msec



Restart: 3 Loglikelihood: -8110.320785645328 Time: 5 sec 799msec

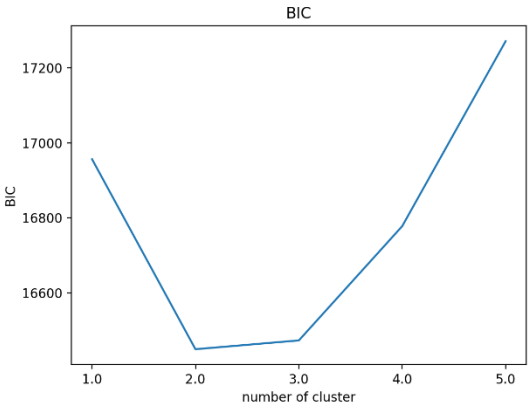


Restart: 5 Loglikelihood: -8110. 31418849481 Time: 7 sec 796msec



The maximum restart time is 6 in provided data set.

- **Search space for number of clusters:** we start from 1 and increase by 1 at each time, until we exhaust the given time. Then we pick the one with best BIC. We show a plotted graph of BIC with respect of number of clusters down below.



4. **Performance:**
   - **Performance on synthetic Data:** We manually created synthetic data and tested the performance of our program. Here we show the two generated output summaries (Test 1 and Test 2). Our program was pretty good at predicting the actual means and variances of the clusters.
   - **Performance on provided Data:** Even though the true cluster number is three, our program chose two as the best cluster number. However, the shape of the original data set is more like two clusters, the BIC of two clusters and three clusters are similar, and BIC penalizes the model with too many parameters. Thus, our program is good enough to select a best number of clusters.

Test 1:

Truth: Two clusters

Mean:[10,10]

Cov: [[16, 1], [1, 9]]

Mean:[15,15]

Cov: [[9, 1], [1, 25]]

Test 1 Result:
Best fitting clusters: Number of clusters: 2

Mean: [10.31429458, 9.93377365]

Cov: [[16.26231053, 1.31175718], [1.31175718, 9.81465029]]

Weight: 0.67064622

Mean: [15.59727807, 15.98018327]

Cov: [[7.60719423, -0.69175412], [-0.69175412, 23.8383522]]

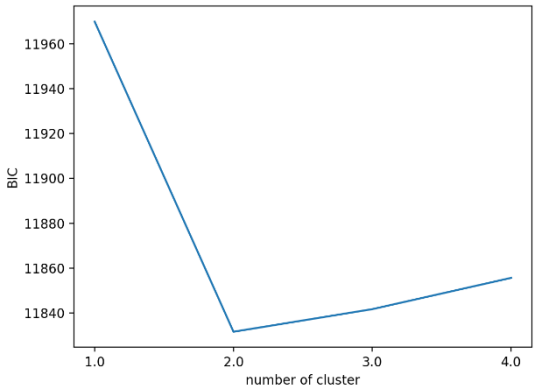
Weight: 0.32935378

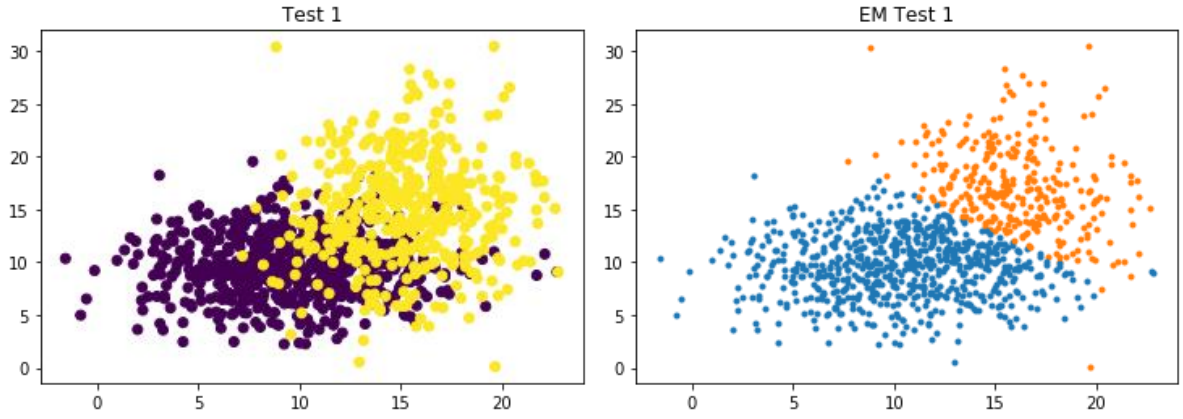
Total Log-likelihood: -5901.927469

Model BIC: 11831.600309

Time: 0.0 hour 0.0 min 3 sec 119 msec

#Restarts: 3





Test2:
Truth: Four clusters
Mean: [2,2]

Cov: [[9, 1], [1, 4]]

Mean: [9,3]

Cov: [[5, 1], [1, 25]]

Mean: [7,10]

Cov: [[16, 1], [1, 16]]

Mean: [14,7]

Cov: [[5, 1], [1, 10]]

Model:
Number of clusters: 4

Mean: [14.13058182 7.46072663]

Cov: [[4.1023647   0.39237395] [0.39237395 11.11354539]]

Weight:0.1782

Mean:[8.97702866 4.56939267]

Cov:[[4.95888877 1.09462795][1.09462795 30.31149115]]

Weight:0.3982

Mean:[1.83610895 1.64229889]

Cov:[[7.90573404 0.75783099][0.75783099 3.07841434]]

Weight:0.2190

Mean:[3.49312165 8.58576148]

Cov: [[9.86599321 6.12512908] [6.12512908 26.73574094]]

Weight:0.2046

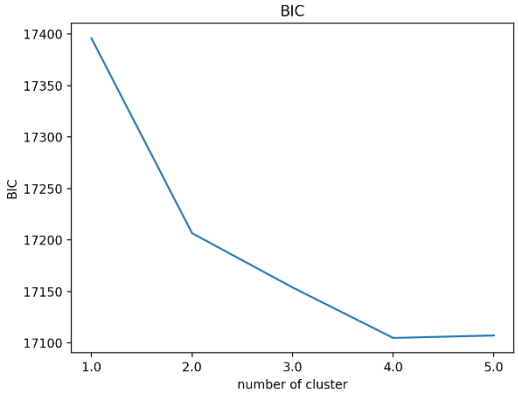
Total Log-likelihood: -8523.4371

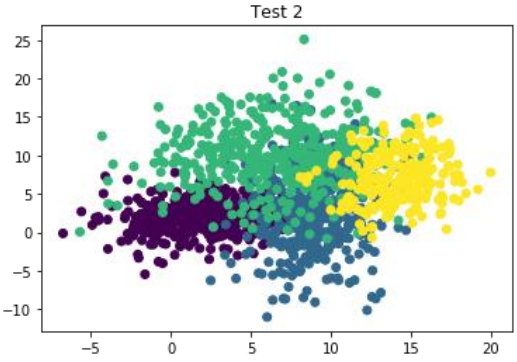
Model BIC: 17105.014419

Simulation statistics:

Time: 0.0 hour 0.0 min 44 sec 328 msec

#Restarts: 3





Provided data:

True mean:

[[9.80297495, 6.37099784],[14.86913925, 50.36962893],[6.34104011, 25.1741941]]

True cov:
[[[16.07226561, -0.78623596], [-0.78623596, 97.00562379]],
[[8.97339439, 0.31751695], [0.31751695, 58.8731658]],
 [[55.92536712, -0.29531562], [-0.29531562, 211.52316194]]]

Model:
Number of clusters: 2

Mean:[14.72556111 49.31573357]

Cov:[[11.6840166   0.33857509][0.33857509 69.29803601]]

Weight: 0.3339

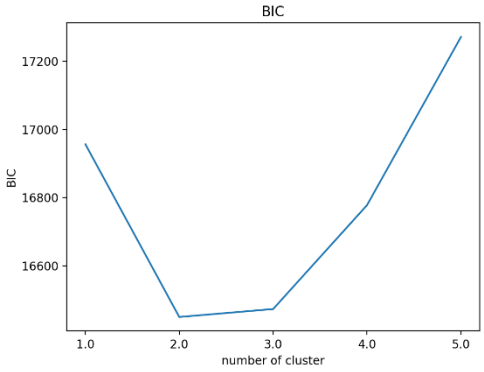
Mean: [8.65347353 9.92753446]

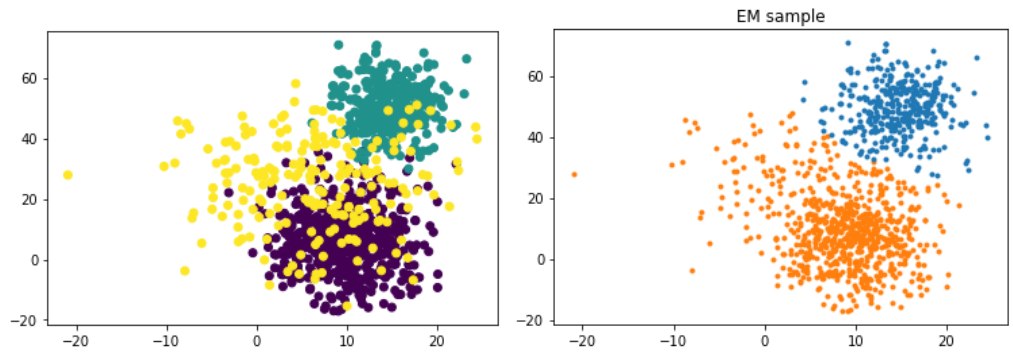
Cov: [[27.63985221 -20.26446953] [-20.26446953 159.97907251]]

Weight: 0.6661

Total Log-likelihood: -8182.8736

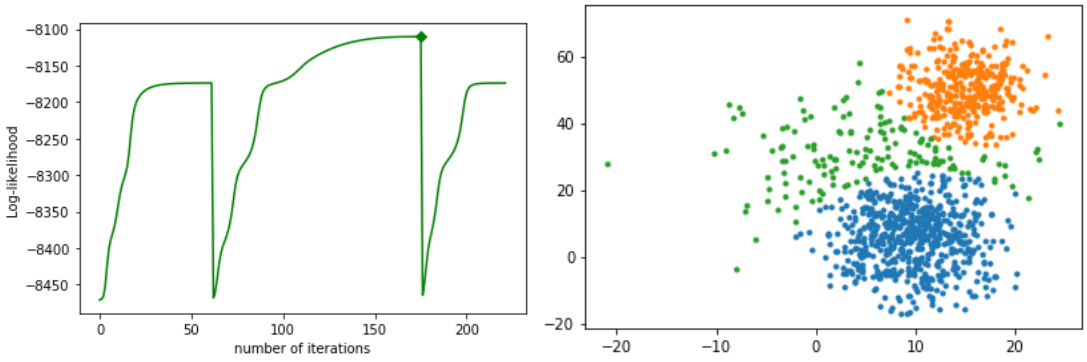
Model BIC: 16449.989577





5. **Log-likelihood VS. #iterations**
    - Down below we plot the log-likelihood vs. #iterations. The sudden drops correspond to the restarts of simulations, and the marked green dot is where the global best stays at.
    - We compared the performance of the convergence of log likelihood and model parameter estimates, which is calculated by the difference between mean and variance. The thresholds are the same.

Log-likelihood with 3 restarts.
convergence of log-likelihood:



Number of clusters: 3

Mean: [14.92803438 50.36860085]

Cov: [[9.41983552 0.30231695] [0.30231695 59.17938948]]

Weight:0.2952

Mean: [6.36959964 29.66239835]

Cov: [[58.87301325 12.07112039] [12.07112039 146.55464349]]

Weight: 0.1721

Mean: [9.72003938 5.82918157]

Cov: [[16.11561251 -0.97349968] [-0.97349968 86.81839527]]

Weight: 0.5327

Total Log-likelihood: -8110.2975

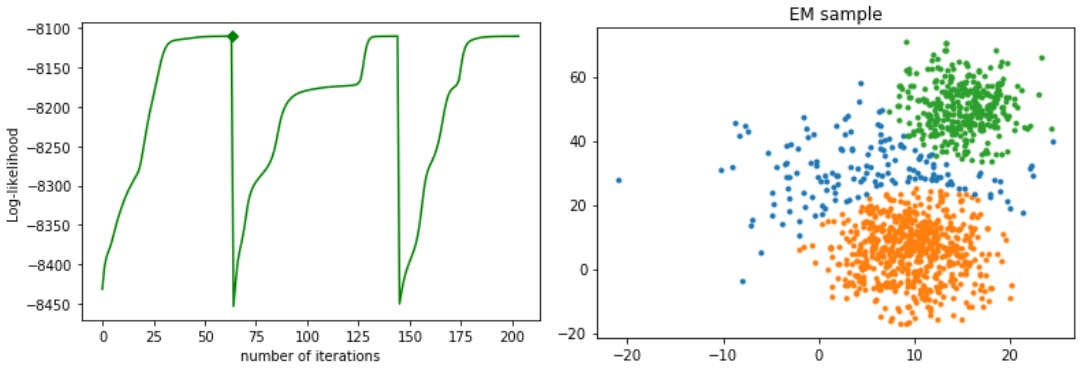
Model BIC: 16473.364304

Time: 0.0 hour 0.0 min 3 sec 980 msec

#Restarts: 3

Later on we kept on doing our experiments, trying to find out the convergence of model parameter estimates. Down below we show our results.

convergence of model parameter estimates:



Number of clusters: 3

Mean:[6.23002182 29.70842272]

Cov:[[59.01889469 11.69926698] [11.69926698 143.05158732]]

Weight:0.1672

Mean: [9.71822899 5.8838185]

Cov: [[16.18621806 -0.92048629] [-0.92048629 87.38441554]]

Weight: 0.5355

Mean: [14.91819619 50.3196086]

Cov: [[9.51486485 0.32976003] [0.32976003 59.70363674]]

Weight: 0.2973

Total Log-likelihood: -8110.2787

Model BIC: 16474.678629

Time: 0.0 hour 0.0 min 6 sec 909 msec

#Restarts: 3