CS 6190: Probabilistic Machine Learning Spring 2022

Homework 4

Handed out: 31 March, 2022
Due: 11:59pm, 15 April, 2022

- You are welcome to talk to other members of the class about the homework. I am more concerned that you understand the underlying concepts. However, you should write down your own solution. Please keep the class collaboration policy in mind.
- Feel free discuss the homework with the instructor or the TAs.
- Your written solutions should be brief and clear. You need to show your work, not just the final answer, but you do *not* need to write it in gory detail. Your assignment should be **no more than 10 pages**. Every extra page will cost a point.
- Handwritten solutions will not be accepted.
- The homework is due by **midnight of the due date**. Please submit the homework on Canvas.

Practice [100 points + 100 bonus]

1. [20 points] Suppose we have a scalar distribution,

$$p(z) \propto \exp(-z^2)\sigma(10z+3).$$

(a) [3 points] Although the normalization constant is not analytical, we can use Gauss-Hermite quadrature to calculate an accurate approximation. Please base on the example in "data/example-code/gmq_example.py", calculate the numerical approximation of the normalization constant, and report its value. With the normalization constant, please draw the density curve of $p(z)$, in the range $z \in [-5, 5]$.

Answer

Code for this question can be found here and I've also implemented this in Jupyter notebook here.

We know that $p(z) \propto \exp(-z^2)\sigma(10z+3)$, however the exact equation can be written as $p(z) = \frac{\exp(-z^2)\sigma(10z+3)}{Z}$. Now to get the normalisation constant we can use numerical approximation as follows:

$$\int_{-\infty}^{\infty} p(z)dz = \int_{-\infty}^{\infty} \exp -z^2\sigma(10z+3)$$

$$\int_{-\infty}^{\infty} p(z)dz \approx \int_{-\infty}^{\infty} w_i\sigma(10z_i+3)$$

Note that in the provided 'gass_hermite_quad' function accepts degree argument and for different values we get different normalisation constants. Here are a few results with different value of degrees.

i. For degree 10, normalisation constant is 1.1262949412781722
ii. For degree 20, normalisation constant is 1.1805811706943297
iii. For degree 50, normalisation constant is 1.1699549724689828
iv. For degree 70, normalisation constant is 1.1685355738566705
v. For degree 100, normalisation constant is 1.16882628660156

Note that for larger degrees, the result is very close hence for the sake of final results, we can consider the degree of 100. The density curve is shown in Figure 1

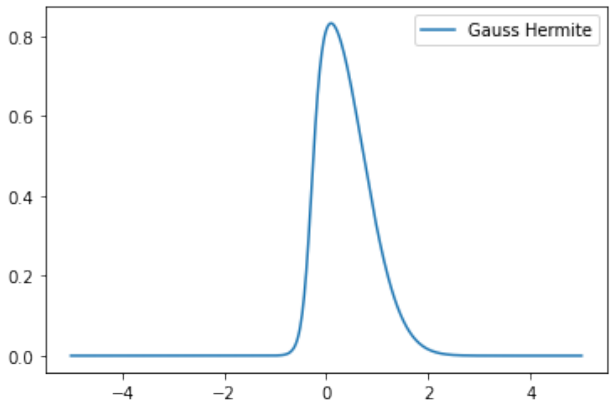

Figure 1: Density curve for Gauss Hermite Quadrature.

(b) [5 points] Implement the Laplace approximation of $p(z)$, and report the mean and variance of your Gaussian distribution. Draw the density of your Laplace approximation in the same plot as in (a).

Answer

The Laplace approximation is implemented by first taking the derivative of $\log(p(z))$ because it helps us get μ. The value at which it gets maximum is the μ. Now let's look at the 2nd order derivative.

$$\frac{d^2}{dz^2}\log(f(z)) = -2 - 100\sigma(10z+3)(1-\sigma(10z+3))$$

Let the value of this second order derivative at $z = \mu$ be A.

$$A = -\frac{d^2}{dz^2}\log(f(z))\Big|_{z=\mu}$$

Now approximating, we get

$$f(z) \approx f(\mu)\exp -\frac{A}{2}(z-\mu)^2$$

Now integrating this gives the normalisation factor

$$Z = f(\mu)\sqrt{\frac{2\pi}{A}}$$

Now plugging this into the original equation gives as the form of a gaussian distribution with mean $\mu = 0.094$ and variance $A^{-1} = 0.25$. The combined density with part a is shown in Figure 2

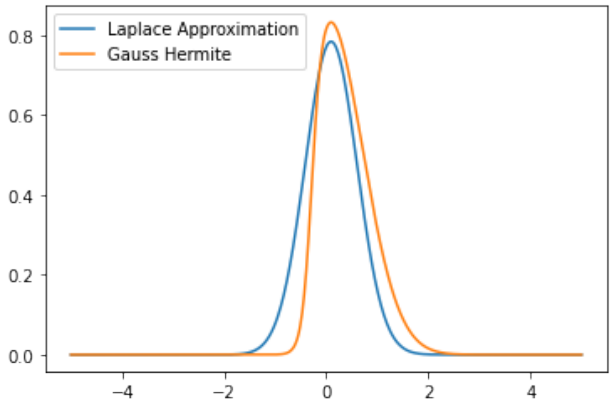

Figure 2: Density curve for Gauss Hermite Quadrature and Laplace Approximation.

(c) [10 points] Use the local variational inference method and EM-style updates as we discussed in the class (for logistic regression) to implement the variational approximation to $p(z)$. Report the form of your approximate distribution, and draw its density in the same plot as above.

Answer

We know that $\sigma(x) \geq \sigma(\psi) \exp\left(\frac{x-\psi}{2} + \lambda(\psi)(x^2 - \psi^2)\right)$

Now transforming the variables x and psi based on the ones provided in the equation with this question we get

$$\exp(-z^2)\sigma(10z+3) \geq \exp(-z^2)\sigma(\chi) \exp\left(\frac{10z+3-\chi}{2} + \lambda(\chi)(100z^2 + 60^2 + 9 - \chi^2)\right)$$

By running multiple updates of EM using the above equation as lower bound. Once we have that lower bound, we maximise z and then we reuse this value as new χ. We keep updating until a threshold is reached. This gives a gaussian distribution as shown in Figure 3

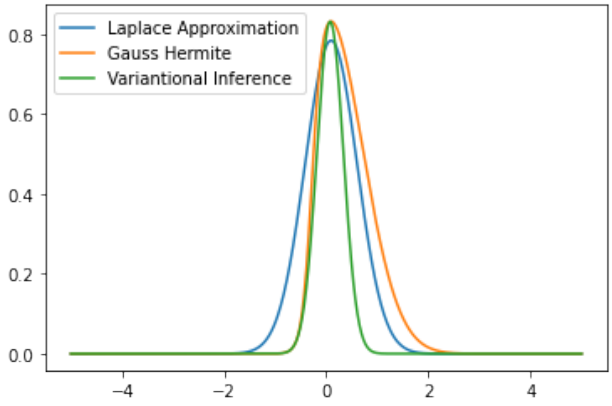

Figure 3: Density curve for Gauss Hermite Quadrature, Laplace Approximation and variational inference.

(d) [2 points] By comparing the "ground-truth" (from (a)) and the approximations (from (b,c)), what do you observe and conclude?

Answer

From above figures we can see that the Laplace Approximation is the worse and the reason being the consideration of local curvature only. Local Variational approximation is the best where it considers global properties along with multiple updates.

2. [50 points] Let us work on a real-world dataset we have met before. Please download the data from the folder"data/bank-note". The features and labels are listed in the file "data-desc.txt". The training data are stored in the file "train.csv", consisting of 872 examples. The test data are stored in "test.csv", and comprise of 500 examples. In both the training and testing datasets, feature values and labels are separated by commas. We assign the feature weight vector $\mathbf{w}$ a standard normal prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

   (a) [7 points] Implement the standard Laplace approximation to the posterior distribution of the feature weight vector. Report your approximate posterior. Now, use Gauss-Hermite quadrature to implement the calculation of the predictive distribution. Please be careful: **you need to do a proper variable transformation in the integral before applying the Gauss-Hermite quadrature because you integrate with a Gaussian like $\mathcal{N}(x|\mu, \sigma^2)$ rather than $\exp(-x^2)$!** Now we test the performance with two measures. First, we calculate the inner-product between the posterior mean of the weight vector and the feature vector of each test example, and throw the inner-product into the sigmoid function to calculate the probability that the test example is positive. If the probability is no less than 0.5, we classify the example to be positive (i.e., 1) otherwise we classify the example to be negative (i.e., 0). Report the prediction accuracy. Second, we calculate the average predictive likelihood of the test samples, namely we evaluate the predictive density value of each test sample based on the predictive distribution and then take an average. Note that in Bayesian learning, the predictive likelihood includes all the information of the (approximate) posterior distribution, hence is more preferred in the evaluation.

   Answer

   Code for this question can be found here and I've also implemented this in Jupyter notebook here.

   To calculate mean and covariance matrix via Laplace approximation, we first use the training data to minimise the log posterior. This yields the mean vector, and to compute covariance matrix, we use this along with the training features. After the calculations, the results are as follows:

$$\mu = \begin{pmatrix} -2.69321794 & -1.59105672 & -1.899262 & -0.17689793 & 2.8559403 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 0.1196425 & 0.05318792 & 0.0718138 & 0.00963916 & -0.05634873 \\ 0.05318792 & 0.05011861 & 0.05299435 & 0.02319023 & -0.02708633 \\ 0.0718138 & 0.05299435 & 0.06217006 & 0.01966921 & -0.04066413 \\ 0.00963916 & 0.02319023 & 0.01966921 & 0.0382288 & 0.02068531 \\ -0.05634873 & -0.02708633 & -0.04066413 & 0.02068531 & 0.13679755 \end{pmatrix}$$

   After having the μ and Σ, we can proceed to the calculation of test accuracy and prediction likelihood. Calculating test accuracy is straight forward and uses only the learned parameters i.e. μ, while predictive likelihood requires both μ and Σ along with gauss hermite approximation (which uses the provided mean and covariance). The results from this are as shown below:

   Test Accuracy: 99%

   Predictive likelihood : 0.9745001197087079

   (b) [3 points] Implement Laplace approximation with the diagonal Hessian. Report the approximate posterior distribution of the feature weights, the prediction accuracy and average predictive likelihood.

   Answer

The only change in this part is that the Σ matrix is converted into a diagonal matrix by zeroing out all off-diagonal elements and then fed to the same functions as used in previous part. The used μ and Σ are as follows:

$$\mu = \begin{pmatrix} -2.69321794 & -1.59105672 & -1.899262 & -0.17689793 & 2.8559403 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 0.1196425 & 0 & 0 & 0 & 0 \\ 0 & 0.05011861 & 0 & 0 & 0 \\ 0 & 0 & 0.06217006 & 0 & 0 \\ 0 & 0 & 0 & 0.0382288 & 0 \\ 0 & 0 & 0 & 0 & 0.13679755 \end{pmatrix}$$

The results are as follows:

Test Accuracy: 99%

Predictive likelihood : 0.9591358360740148

(c) [20 points] Implement variational logistic regression we introduced in the class. Use EM-style updates. Report the variational posterior of the feature weight vector you obtained (i.e., a multivariate Gaussian). Report the prediction accuracy and average predictive likelihood.

Answer

Variational Logistic regression computes μ and Σ matrix differently. I used 100 updates to refine the parameteres and here are the results.

$$\mu = \begin{pmatrix} -2.50491512 & -1.46827861 & -1.75477872 & -0.13286228 & 2.79364586 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 0.00328581 & -0.00021109 & 0.0007029 & -0.00176941 & -0.0026148 \\ -0.00021109 & 0.00171986 & 0.00143007 & 0.00152302 & -0.00186669 \\ 0.0007029 & 0.00143007 & 0.00210823 & 0.00037659 & -0.00398946 \\ -0.00176941 & 0.00152302 & 0.00037659 & 0.00592795 & 0.00478958 \\ -0.0026148 & -0.00186669 & -0.00398946 & 0.00478958 & 0.02726292 \end{pmatrix}$$

The results are as follows:

Test Accuracy: 99%

Predictive likelihood : 0.9750105165467248

(d) [15 points] Implement variational logistic regression we introduced in the class. But this time, you will use the fully factorized posterior, $q(\mathbf{w}) = \prod_i q(w_i)$ where w_i is i-th element in the weight vector $\mathbf{w}$. In the E step, please use the standard mean-field update to alternatively optimize each $q(w_i)$ given all the others fixed. Report your variational posterior (i.e., diagonal Gaussian), the prediction accuracy and average predictive likelihood on the test data.

Answer

The results for this part are shown as below.

$$\mu = \begin{pmatrix} -1.95838294 & -1.01324036 & -1.24271807 & 0.03999042 & 2.43016352 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 0.00184179 & 0. & 0. & 0. & 0. \\ 0. & 0.0003503 & 0. & 0. & 0. \\ 0. & 0. & 0.00048542 & 0. & 0. \\ 0. & 0. & 0. & 0.002103720. & \\ 0. & 0. & 0. & 0. & 0.01161653 \end{pmatrix}$$

The results are as follows:

Test Accuracy: 98.4%

Predictive likelihood : 0.9666392736213947

(e) [5 points] Compare the results of the above four approximations. What do you observe and conclude?

Answer

All the above methods have almost same accuracy with variational logictic regression using mean field inference being slightly poor. The main point to note is that the off diagonal elements are smaller as compared to the diagonal ones hence the fully factorised approximation is not that bad. I think this was the reason for accuracy being only slightly worse when using only the diagonal elements in case of variational logistic regression. Another difference is in the likelihood results where I think variational logistic inferecne does better and can handle complex datasets well.

3. [30 points] Gaussian Mixture Model (GMM). Please download the data "data/faithful/faithful.txt" from Canvas. Each row is a sample, including 2 features. Please normalize the features in each column to be in [-1, 1]. Specifically, denote the column by $\mathbf{x}$; then we compute for each $x_i \leftarrow (x_i - \text{mean}(\mathbf{x}))/(\max(\mathbf{x}) - \min(\mathbf{x}))$.

   (a) [20 points] Implement EM algorithm for GMM. Set the number of clusters to 2. Initialize the cluster centers to be [-1, 1] and [1,-1], and the covariance matrix to be $0.1 \cdot \mathbf{I}$ for both clusters. Run your EM algorithm for 100 iterations. For iteration 1, 2, 5, 100, please draw the figures showing the corresponding cluster centers and memberships. Specifically, for each figure, first draw the scatter plots of the data points and your cluster centers. Each data point is assigned to the cluster that has a great posterior probability to include that data point. Please draw the cluster memberships with different colors.

   Answer

   Code for this question can be found here and I've also implemented this in Jupyter notebook here.

   For results please see Figure 4

   (b) [7 points] Now initialize the cluster centers to be [-1, -1] and [1, 1] and covariance matrix to be $0.5 \cdot \mathbf{I}$ for both clusters. Run your EM algorithm for 100 iterations. Draw the figures showing the cluster centers and memberships for iteration 1, 2, 5, 100.

   Answer

   For results please see Figure 5

   (c) [3 points] Compare the results in (a) and (b), what do you observe and conclude?

   Answer

   From the results in previous parts we can see that the initialisation is very important. When we initialise in part a, it takes some time for model to converge however in part b, it converged quickly.

   For the results where there's no convergence, please see Figure 6

4. [100 points][**Bonus**] Latent Dirichlet Allocation (LDA). Please download the pre-processed corpus from "data/lda". From "ap.txt", you can see the original Associated Press corpus. "ap.dat" are the processed data which you will work on. Each row in "ap.dat" represents a document. In total, we have $2,246$ documents. The first number in each row is the number of words in the document. The following are a list of **word-id**:**count** items. Word-id starts with 0. The corresponding word list is given in "vocab.txt". The first row correspond to Word-id 0, second, Word-id 1, and continue.

   (a) [70 points] Implement the mean-field variational EM algorithm for LDA inference as we discussed in the class. Following the orignal LDA paper (`http://www.cs.columbia.edu/~blei/papers/BleiNgJordan2003.pdf`) to implement the perplexity calculation on test documents (Sec. 7.1). Please randomly select 10% documents as the test set, and run your LDA inference algorithm on the remaining 90% documents. Vary the number of topics from {5, 10, 15, 20, 50, 100, 200}. Run your algorithm until convergence or 500 iterations have achieved. Draw a

figure to show how the perplexity vary along with the growth of the topic numbers. What do you observe and conclude?

(b) [30 points] Set the number of topics to 20 and run your variational inference algorithm. Examine the top 15 words (i.e., with the largest probability) in each learned topic distribution. List a few topics which you think is semantically meaningful and explain why.

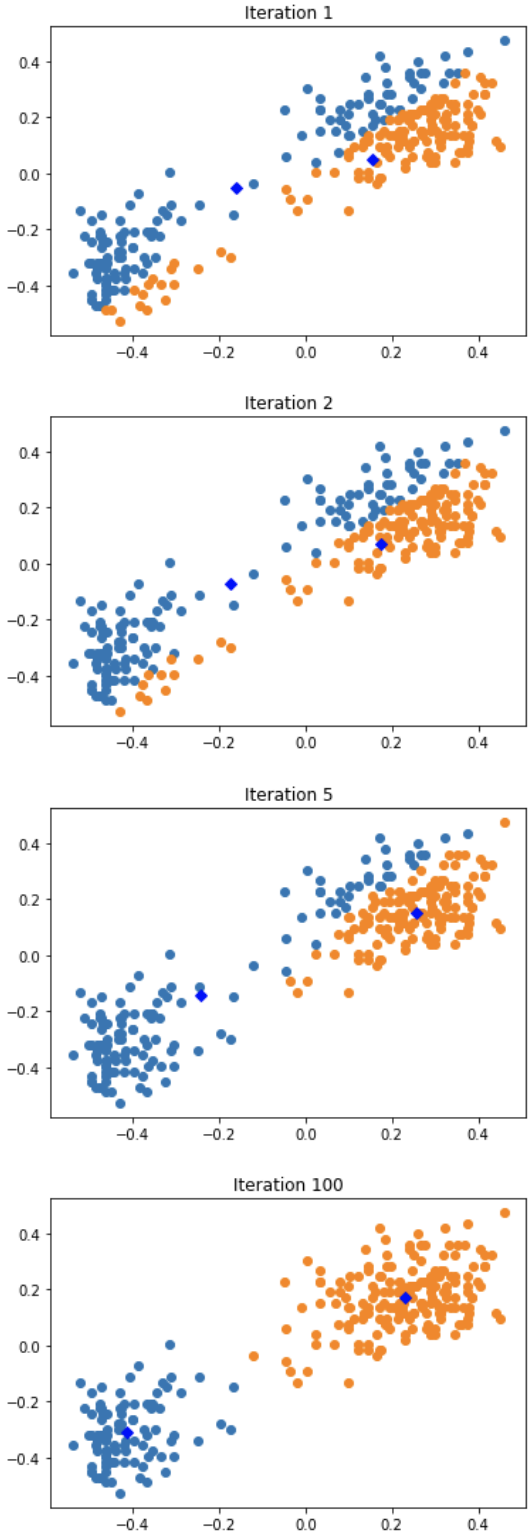

Figure 4: Cluster after several updates when center is [-1,1] and [1,-1], and covariance matrix is 0.1*I.

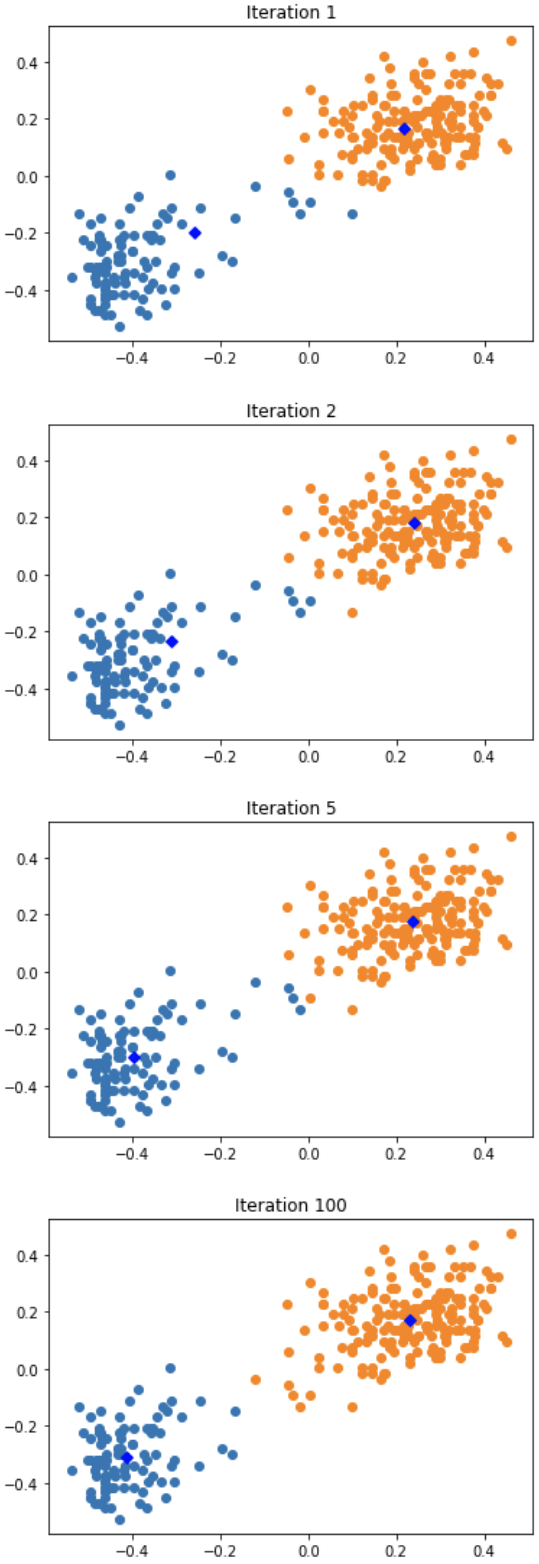

Figure 5: Cluster after several updates when center is [-1,-1] and [1,1], and covariance matrix is 0.5*I.

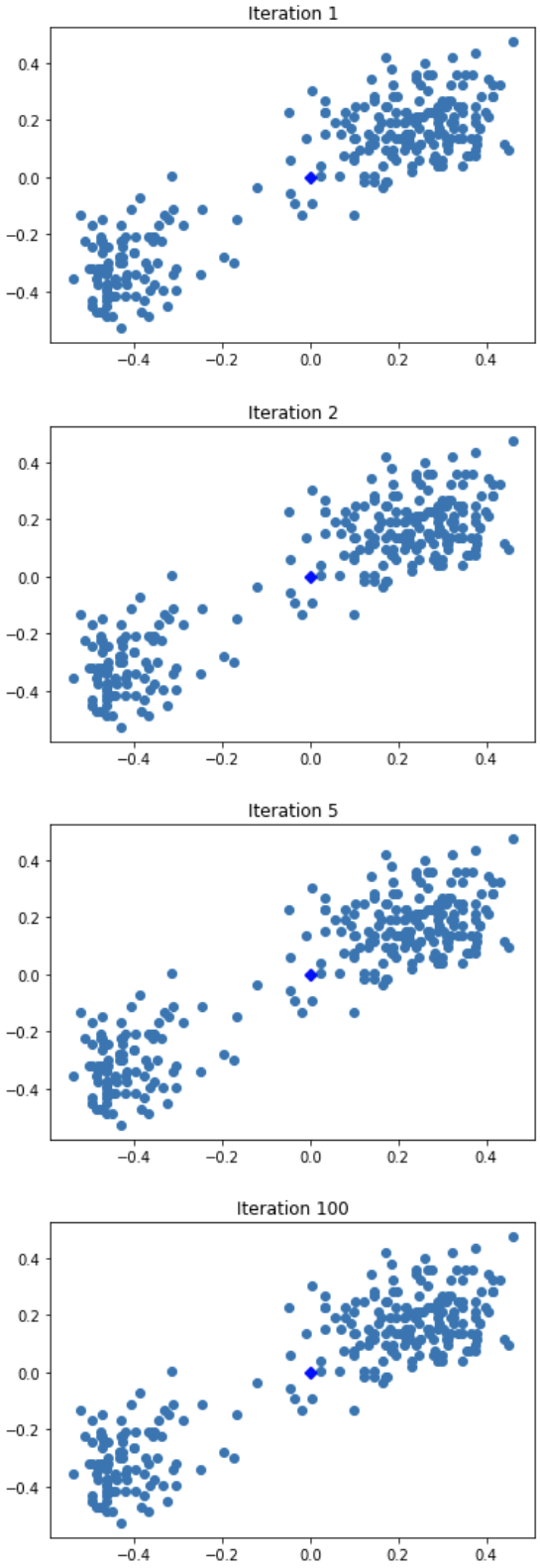

Figure 6: Cluster after several updates when center is [0,0] and [0,0], and covariance matrix is 0.5*I.