CS 6190: Probabilistic Modelling Spring 2019

Homework 4

Abhinav Kumar (u1209853)

November 22, 2019

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 $z \in [-5, 5]$.

With degree or number of sampled points = 100, we get the normalization constant as 1.16

(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).

The double derivative of $\ln(z)$ is given by $2 + 100\sigma(10z+3)(1-\sigma(10z+3))$. Mean = 0.09 and variance = 3.94. The plot is shown in Figure 1.

(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.

We use the local variational inference discussed in the class. We derived that

$$\sigma(x) \geq \sigma(\psi)e^{\frac{1}{2}(x-\psi)+\lambda(\psi)(x^2-\psi^2)}$$

Transforming x to z and ψ to ξ by the transformation, $x = 10z+3$ and $\psi = 10\xi+3$, we get

$$\sigma(10z+3) \geq \sigma(10\xi+3)e^{5(z-\xi)+\lambda_{(10\xi+3)} \quad 10(z-\xi)(10z+10\xi+6)}$$

where $\lambda_{(10\xi+3)} = \frac{-1}{2(10\xi+3)}\left[\sigma(10\xi+3) - 0.5\right]$

Hence,

$$p(z) \propto \exp(-z^2)\sigma(10z+3) \geq \exp(-z^2)\sigma(10\xi+3)e^{5(z-\xi)+\lambda_{(10\xi+3)} \quad 10(z-\xi)(10z+10\xi+6)}$$

We run multiple updates of EM using the above lower bound. Once we obtain the lower bound, we maximize for the value of z which becomes our new ξ. Then, we check the difference between the older estimate and the newer estimate of ξ to decide whether we proceed with the next iteration or not.

Clearly, this is a normal distribution with mean= 0.09 and variance= 0.22. The plot is shown in Figure 1.

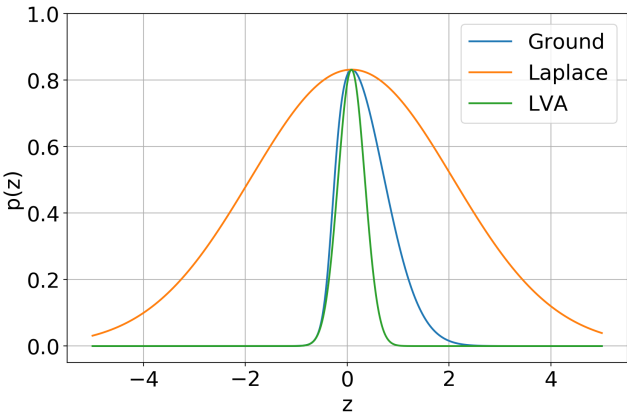


Figure 1: True distribution (Ground), Laplace approximation and its Local Variational Approximation (LVA).

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

Laplace approximation is not a good approximation of the true distribution since it only takes the local curvature at the mode into account. Local variational approximation, on the other hand, approximates the function well since it uses the global properties (convex dual) of transformed sigmoid function and also runs multiple updates and hence is more accurate.

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.

   We want to minimize the posterior $-\ln p(\mathbf{w}|t) = -\ln p(t|\mathbf{w}) - \ln p(\mathbf{w})$.

$$\mathbb{E}(-\ln p(t|\mathbf{w}) - \ln p(\mathbf{w})) = -\sum_{n=1}^{N} t_n \ln y_n + (1-t_n)\ln(1-y_n) + \ln(\lambda) + \frac{1}{2}\mathbf{w}^T\mathbf{w} \tag{1}$$

   where $y_n = \sigma(\mathbf{w}^T\boldsymbol{\phi}_n)$.

   Differentiating once wrt $\mathbf{w}$, we have

$$\begin{aligned}\nabla_{\mathbf{w}}\mathbb{E} = \nabla_{\mathbf{w}}\mathbb{E}(-\ln p(t|\mathbf{w}) - \ln p(\mathbf{w})) &= -\sum_{n=1}^{N} \frac{t_n}{y_n} y_n(1-y_n)\boldsymbol{\phi}_n - \frac{(1-t_n)}{(1-y_n)} y_n(1-y_n)\boldsymbol{\phi}_n + \mathbf{w} \\ &= \sum_{n=1}^{N}(y_n - t_n)\boldsymbol{\phi}_n + \mathbf{w} \\ &= \boldsymbol{\Phi}^T(\mathbf{y} - \mathbf{t}) + \mathbf{w} \end{aligned} \tag{2}$$

   where $\boldsymbol{\Phi}$ is a $n \times d$ matrix and each $\boldsymbol{\phi}_n$ is a row vector.

   Differentiating once more wrt $\mathbf{w}$ to get the Hessian, we have

$$\begin{aligned}\mathbf{S} = \nabla^2_{\mathbf{w}}\mathbb{E} = \nabla^2_{\mathbf{w}}\mathbb{E}(-\ln p(t|\mathbf{w}) - \ln p(\mathbf{w})) &= \nabla_{\mathbf{w}}.\nabla^T_{\mathbf{w}}\mathbb{E}(-\ln p(t|\mathbf{w}) - \ln p(\mathbf{w})) \\ &= \sum_{n=1}^{N} y_n(1-y_n)\boldsymbol{\phi}_n\boldsymbol{\phi}_n^T + \mathbf{I} \\ \text{or, } \mathbf{S} &= \boldsymbol{\Phi}^T\mathbf{R}\boldsymbol{\Phi} + \mathbf{I} \end{aligned} \tag{3}$$

   where $\mathbf{R} = n \times n$ diagonal matrix with $\mathbf{R}_{nn} = y_n(1-y_n)$.

   Now, the predictive likelihood for a single example $\boldsymbol{\phi}$ is

$$\begin{aligned} p(y|\boldsymbol{\phi}, D) &= \int p(y, \mathbf{w}|\boldsymbol{\phi}, D)d\mathbf{w} \\ &= \int p(y|\boldsymbol{\phi}, \mathbf{w})p(\mathbf{w}|D)d\mathbf{w} \end{aligned}$$

$$= \int \sigma(\mathbf{w}^T\boldsymbol{\phi})^t(1-\sigma(\mathbf{w}^T\boldsymbol{\phi}))^{1-t}p(\mathbf{w}|D)d\mathbf{w}$$
$$\approx \int \sigma(\mathbf{w}^T\boldsymbol{\phi})^t(1-\sigma(\mathbf{w}^T\boldsymbol{\phi}))^{1-t}q(\mathbf{w}|D)d\mathbf{w}$$

Without loss of generality, we assume that the example has true label $t = 1$, this reduces to

$$p(y=1|\boldsymbol{\phi}, D) \approx \int \sigma(\mathbf{w}^T\boldsymbol{\phi})q(\mathbf{w}|D)d\mathbf{w}$$
$$= \int \sigma(a)\left(\int \delta(a-\mathbf{w}^T\boldsymbol{\phi})q(\mathbf{w})d\mathbf{w}\right)da$$

The term in brackets is a new distribution $r(a)$ which is a Gaussian since $q(\mathbf{w})$ is a Gaussian owing to the Laplace transformation $q(\mathbf{w}|D) = \frac{C}{|\mathbf{S}|^{0.5}}exp^{-\frac{1}{2}(\mathbf{w}-\mathbf{w}_{MAP})^T\mathbf{S}^{-1}(\mathbf{w}-\mathbf{w}_{MAP})}$.

We therefore use mean and variance to fully quantify $r(a)$. We first calculate mean by interchanging the order of integration over a and w, so that

$$\mathbb{E}_a = \int ar(a)da$$
$$= \int q(\mathbf{w})\int a\delta(a-\mathbf{w}^T\boldsymbol{\phi})dad\mathbf{w}$$
$$= \int q(\mathbf{w})\mathbf{w}^T\boldsymbol{\phi}d\mathbf{w}$$
$$= \mathbf{w}_{MAP}^T\boldsymbol{\phi}$$

Similarly, $Var_a = \boldsymbol{\phi}^T\mathbf{S}\boldsymbol{\phi}$. Hence, we have

$$p(y=1|\boldsymbol{\phi}, D) = \int \sigma(a)\mathcal{N}(a|\mathbf{w}_{MAP}^T\boldsymbol{\phi}, \boldsymbol{\phi}^T\mathbf{S}\boldsymbol{\phi})da$$

Substituting $\frac{1}{\sqrt{2}}\frac{a-\mathbf{w}_{MAP}^T\boldsymbol{\phi}}{\sqrt{\boldsymbol{\phi}^T\mathbf{S}\boldsymbol{\phi}}} = \mathbf{z}$, we have

$$\boxed{p(y=1|\boldsymbol{\phi}, D) = \frac{1}{\sqrt{\pi}}\int \sigma\left(\sqrt{2\boldsymbol{\phi}^T\mathbf{S}\boldsymbol{\phi}}z + \mathbf{w}_{MAP}^T\boldsymbol{\phi}\right)exp(-z^2)d\mathbf{z}}$$

. The prediction likelihood is then given by

$$\boxed{p(y=1|\boldsymbol{\phi}, D)^t\left(1-p(y=1|\boldsymbol{\phi}, D)\right)^{1-t}} \quad (4)$$

Approximate posterior is $\mathcal{N}(\mathbf{w}_{MAP}, \mathbf{S})$
where $\mathbf{w}_{MAP} = [-2.6, -1.5, -2., -0.05, 2.7]$
and $\mathbf{S} = \begin{bmatrix} 104.96 & 197.47 & -301.93 & 27.28 & -21.35 \\ 197.47 & 752.52 & -864.72 & -7.87 & -48.86 \\ -301.93 & -864.72 & 1131.37 & -54.76 & 80.43 \\ 27.28 & -7.87 & -54.76 & 68.83 & -17.10 \\ -21.35 & -48.86 & 80.42 & -17.10 & 18.52 \end{bmatrix}$
Prediction accuracy = 97.8 %
Prediction Likelihood = 0.52

(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.
In Diagonal Laplace approximation, we use only the diagonal terms of the Hessian matrix. Refer A scalable Laplace Approximation of Neural Networks, Ritter et al, ICLR 2018, available at

https://openreview.net/pdf?id=Skdvd2xAZ

Approximate posterior is $\mathcal{N}(\mathbf{w}_{MAP}, \mathbf{S})$
where $\mathbf{w}_{MAP} = [-2.6, -1.5, -2., -0.05, 2.7]$
and $\mathbf{S} = \begin{bmatrix} 104.96 & 0 & 0 & 0 & 0 \\ 0 & 752.52 & 0 & 0 & 0 \\ 0 & 0 & 1131.37 & 0 & 0 \\ 0 & 0 & 0 & 68.83 & 0 \\ 0 & 0 & 0 & 0 & 18.52 \end{bmatrix}$
Prediction accuracy = 97.8 %
Prediction Likelihood = 0.51

(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.

For variational logistic regression with prior as $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The update equations are the ones that were derived in the class.

$$\mathbf{S}_N = \left(\mathbf{S}_0^{-1} + 2 \sum_{n=1}^{N} \lambda(\xi_n) \boldsymbol{\phi}_n \boldsymbol{\phi}_n^T \right)^{-1} \tag{5}$$

$$\mathbf{m}_N = \mathbf{S}_N \left(\sum_{n=1}^{N} (t_n - 0.5) \boldsymbol{\phi}_n \right) \tag{6}$$

$$\xi_n = \sqrt{\boldsymbol{\phi}_n^T \left(\mathbf{S}_N + \mathbf{m}_N \mathbf{m}_N^T \right) \boldsymbol{\phi}_n} \tag{7}$$

where $\lambda(\xi_n) = \frac{1}{2\xi_n}(\sigma(\xi_n) - 0.5)$.

Approximate posterior is $\mathcal{N}(\mathbf{m}_N, \mathbf{S}_N)$
where $\mathbf{m}_N = [-1.64, -0.88, -1.09, 0.06, 2.46]$
and $\mathbf{S}_N = \begin{bmatrix} 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.02 \end{bmatrix}$
Prediction accuracy = 98.4 %
Prediction Likelihood = 0.96

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

For variational logistic regression with prior as $\mathcal{N}(\mathbf{0}, \mathbf{I})$, the lower bound is given by

$$\begin{aligned} ln(q) &\propto -\frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{n=1}^{N} \left(\mathbf{w}^T \boldsymbol{\phi}_n (t_n - 0.5) - \lambda(\xi_n) \mathbf{w}^T (\boldsymbol{\phi}_n \boldsymbol{\phi}_n^T) \mathbf{w} \right) \\ &= -\frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{n=1}^{N} \left(\mathbf{w}^T \boldsymbol{\phi}_n (t_n - 0.5) - \lambda(\xi_n) (\mathbf{w}^T \boldsymbol{\phi}_n)^2 \right) \\ &= -\frac{1}{2} \sum_{k=1}^{d} w_k^2 + w_j \sum_{n=1}^{N} \left(\boldsymbol{\phi}_{nj} (t_n - 0.5) \right) + \sum_{\substack{k=1 \\ k \neq j}}^{d} w_k \sum_{n=1}^{N} \left(\boldsymbol{\phi}_{nk} (t_n - 0.5) \right) - \sum_{n=1}^{N} \lambda(\xi_n) (\sum_{k=1}^{d} w_k \phi_{nk})^2 \end{aligned}$$

$$= -\frac{1}{2}w_j^2 - \frac{1}{2}\sum_{\substack{k=1\\k\neq j}}^{d} w_k^2 + w_j \sum_{n=1}^{N}\left(\boldsymbol{\phi}_{nj}(t_n - 0.5)\right) + \sum_{\substack{k=1\\k\neq j}}^{d} w_k \sum_{n=1}^{N}\left(\boldsymbol{\phi}_{nk}(t_n - 0.5)\right)$$
$$- w_j^2 \sum_{n=1}^{N}\lambda(\xi_n)\phi_{nj}^2 - 2w_j \sum_{n=1}^{N}\lambda(\xi_n)\phi_{nj}\sum_{\substack{k=1\\k\neq j}}^{d} w_k\phi_{nk} + C$$
$$= -w_j^2\left[\frac{1}{2} + \sum_{n=1}^{N}\lambda(\xi_n)\phi_{nj}^2\right] + w_j\left[\sum_{n=1}^{N}(t_n - 0.5)\boldsymbol{\phi}_{nj} - 2\sum_{n=1}^{N}\lambda(\xi_n)\phi_{nj}\left(\sum_{\substack{k=1\\k\neq j}}^{d} w_k\phi_{nk}\right)\right] + C$$

Now, we take expectation wrt all other variables other than w_j. Since, all $w_k \sim N(m_k, \sigma_k^2)$, so we have $\mathbb{E}(w_k) = m_k$ and $\mathbb{E}(w_k^2) = m_k^2 + \sigma_k^2$. Hence,

$$\mathbb{E}_{\neg w_j} ln(q) = -w_j^2\left[\frac{1}{2} + \sum_{n=1}^{N}\lambda(\xi_n)\phi_{nj}^2\right] + w_j\left[\sum_{n=1}^{N}(t_n - 0.5)\boldsymbol{\phi}_{nj} - 2\sum_{n=1}^{N}\lambda(\xi_n)\phi_{nj}\left(\sum_{\substack{k=1\\k\neq j}}^{d} m_k\phi_{nk}\right)\right] + C \tag{8}$$

For fully factorized posterior, the individual $w_j \sim \mathcal{N}(m_j, \sigma_j^2)$ and therefore

$$ln(q_j) \propto -\frac{(w_j - m_j)^2}{2\sigma_j^2}$$
$$= -\frac{w_j^2}{2\sigma_j^2} + w_j\frac{m_j}{\sigma_j^2} + C \tag{9}$$

Comparing (8) with (9), we get the update equations

$$\mathbf{S}_{N_{jj}} = \sigma_j^2 = \left(1 + 2\sum_{n=1}^{N}\lambda(\xi_n)\phi_{nj}^2\right)^{-1} \tag{10}$$

$$m_j = \sigma_j^2\left[\sum_{n=1}^{N}(t_n - 0.5)\phi_{nj} - 2\sum_{n=1}^{N}\lambda(\xi_n)\phi_{nj}\left(\sum_{\substack{k=1\\k\neq j}}^{d} m_k\phi_{nk}\right)\right] \tag{11}$$

$$\xi_n = \sqrt{\boldsymbol{\phi}_n^T\left(\mathbf{S}_N + \mathbf{m}_N\mathbf{m}_N^T\right)\boldsymbol{\phi}_n} \tag{12}$$

where $\lambda(\xi_n) = \frac{1}{2\xi_n}(\sigma(\xi_n) - 0.5)$.

Approximate posterior is $\mathcal{N}(\mathbf{m}_N, \mathbf{S}_N)$
where $\mathbf{m}_N = [-1.56, -0.69, -0.91, 0.13, -2.21]$
and $\mathbf{S}_N = \begin{bmatrix} 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.01 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.01 \end{bmatrix}$
Prediction accuracy = 98.00 %
Prediction Likelihood = 0.95

Note:

- I also tried without inplace update (using older means in successive epochs the new means). The convergence point reached is same. It only took one more iteration.

- The equations (10) can be obtained if we use only the diagonal terms of (5). However, (11) can not be obtained in the same manner from (6).

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

- All the methods perform very close to each other in the sense of prediction accuracy close to 98%. However, out of them Variational Logistic Regression performs the best.
- The prediction likelihood tells us how well the modelling is done. The order is
  Full Variational > Factorised Variational > Full Laplace > Diagonal Laplace

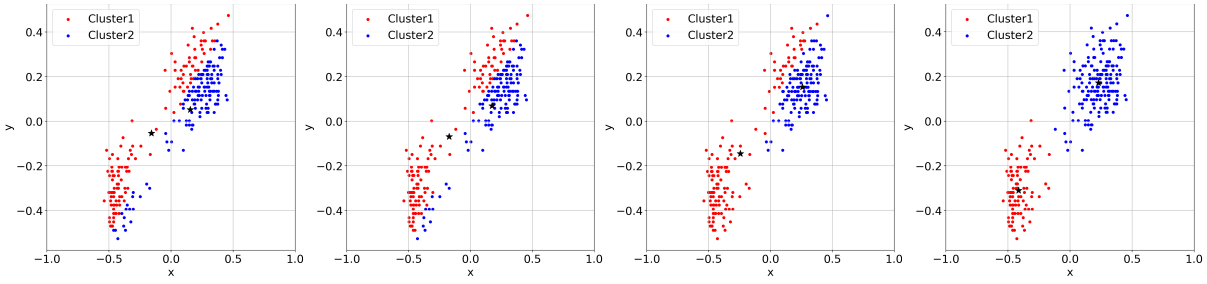
  This is expected since the Full Variational is based on the convex dual properties of the transformed sigmoid and also uses multiple updates to reach the optimum. The performance of Factorised Variational is slightly low in terms of prediction likelihood owing to the fact that the former uses only diagonal covariance matrices.

  Laplace Approximation performs inferior compared to the Variational Approximations since Laplace approximation uses local curvature information around the MAP point and also uses single update to get the approximation. Hence, the posterior obtained is not that good. As was the case of Factorised Variational vs Full Variational, Diagonal Laplace performs slightly inferior in terms of prediction likelihood as compared to Full Laplace.
- Full Variational fits the data better than the Factorised Variational since the former uses full covariance matrices. The drawback lies in the fact that it requires inversion of the covariance matrices and therefore can be expensive to compute for high dimensional data.

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.

   I referred to the GMM EM equations listed at https://www.ics.uci.edu/~smyth/courses/cs274/notes/EMnotes.pdf. for my code. The plots are shown in Figure 2

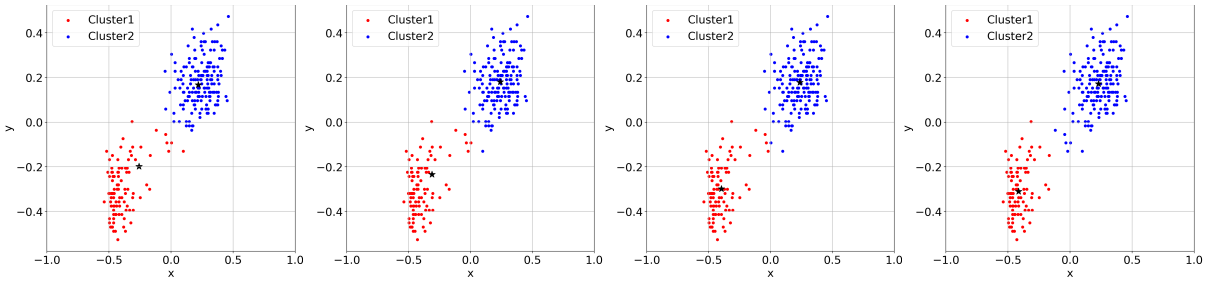


(a) Iteration 1 (b) Iteration 2 (c) Iteration 5 (d) Iteration 100

Figure 2: Training GMM

   (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.

   The plots are shown in Figure 3.



(a) Iteration 1 (b) Iteration 2 (c) Iteration 5 (d) Iteration 100

Figure 3: Training GMM

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

   We notice that the seeding or the initialization point controls the speed of convergence. In 3(a), the seeding points were very poorly choosen (much far from the true mean) and hence it took 100 steps to converge. In 3(b), the seeding points for the mean were very close to the true mean and therefore GMM training converged in 5 steps.