Jashaswimalya Acharjee
CS22E005
October 25, 2022

CS 5691: Pattern Recognition and Machine Learning
Assignment: 2
Course Instructor : Arun Rajkumar

Question: 1

You are given a data-set with 400 data points in $\{0,1\}^{50}$ generated from a mixture of some distribution in the file 'A2Q1.csv'. (Hint: Each datapoint is a flattened version of a $\{0,1\}^{10\times 5}$ matrix.)

(i): Determine which probabilisitic mixture could have generated this data (It is not a Gaussian mixture). Derive the EM algorithm for your choice of mixture and show your calculations. Write a piece of code to implement the algorithm you derived by setting the number of mixtures $K = 4$. Plot the log-likelihood (averaged over 100 random initializations) as a function of iterations.

Ans: It can be concluded by me, after plotting each datapoint of $\{0,1\}^{1\times 50}$ as well as visualizing the sum along each axis, that the given datapoints have been sampled from a Mixture of Bernoulli — with success or failure characteristic feature, i.e. (0,1).

Derivation: The mixture of 'n' discrete binary variables described by Bernoulli Distributions can be modeled with EM as *Latent Class Analysis*.

Consider a set of D binary variables x_i, where $i = 1, 2, \ldots, D$, each of which is governed by a Bernoulli Distribution with some parameter μ_i, so that

$$p(x|\mu) = \Pi_{i=1}^{D} \mu_i^{x_i}(1-\mu_i)^{(1-x_i)}$$

The mean and covariance of this distribution are μ and $diag\{\mu_i(1-\mu_i)\}$. Hence, the finite mixture of these distributions can be expressed as:

$$p(x|\mu,\pi) = \sum_{k=1}^{K} \pi_k p(x|\mu_k) \quad \text{and} \quad p(x|\mu_k) = \Pi_{i=1}^{D} \mu_{ki}^{x_i}(1-\mu_{ki})^{(1-x_i)}$$

Henceforth we redefine the mean and covariance of this mixture distribution as:

$$\mathbb{E}[x] = \sum_{k=1}^{K} \pi_k \mu_k \quad \text{and} \quad \sum_{k=1}^{K} \pi_k\{\Sigma_k + \mu_k \mu_k^T\} - \mathbb{E}[x]\mathbb{E}[x]^T$$

respectively, where $\Sigma_k = diag\{\mu_{ki}(1-\mu_{ki})\}$.

To derive the EM for this mixture, let $\mathbf{z}$ be the latent variable associated with each instance of $\mathbf{x}$. Therefore, the conditional distribution of $\mathbf{x}$, given the latent variable $\mathbf{z}$ can be written as:

$$p(x|z,\mu) = \Pi_{k=1}^{K} p(x|\mu_k)^{z_k}$$

with the prior distribution for the latent variables as

$$p(z|\pi) = \Pi_{k=1}^{K} \pi_k^{z_k}$$

Taking the expectation of the log-likelihood with respect to posterior distribution of the latent variables, yields:

$$\mathbb{E}_z[\ln p(X,Z|\mu,\pi)] = \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma(z_{nk}) \Big\{ \ln \pi_k + \sum_{i=1}^{D} [x_{ni} \ln \mu_{ki} + (1-x_{ni}) \ln(1-\mu_{ki})] \Big\}$$

where $\gamma(z_{nk}) = \mathbb{E}[z_{nk}]$ is the posterior probability of component k given data point x_n.

In the **E** step, these posterior probabilities are evaluated using Bayes' theorem, which takes the form

$$\gamma(z_{nk}) = \mathbb{E}[z_{nk}] = \frac{\sum_{z_{nk}} z_{nk}[\pi_k p(x_n|\mu_k)]^{z_{nk}}}{\sum_{z_{nj}}[\pi_j p(x_n|\mu_j)]^{z_{nj}}}$$
$$= \frac{\pi_k p(x_n|\mu_k)}{\sum_{j=1}^{K} \pi_j p(x_n|\mu_j)}$$

If we consider the sum over n in the Loglikelihood expression, we can obtain the posterior probabilities enter only through two terms:

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}) \quad \text{and} \quad \bar{x_k} = \frac{1}{N-k}\sum_{n=1}^{N} \gamma(z_{nk}x_n)$$

where, N_k is the effective number of datapoints associated with component k.

In the M step, we maximize the expected complete-data loglikelihood with respect to the parameters μ_k and π. If we set the derivative of the initial Loglikelihood Expression with respect to μ_k equal to zero (0), we obtain:

$$\mu_k = \bar{x_k}$$

For the maximization with respect to π_k, we need to introduce a Lagrange Multiplier to enforce the constraint $\sum_k \pi_k = 1$, thus following steps analogous to the Mixture of Gaussians,

$$\pi_k = \frac{N_k}{N}$$

CODE: Attached in the specified format as .py files. Plot for the aforementioned expression of loglikelihood averaged over 100 random initializations can be seen from Fig: 1

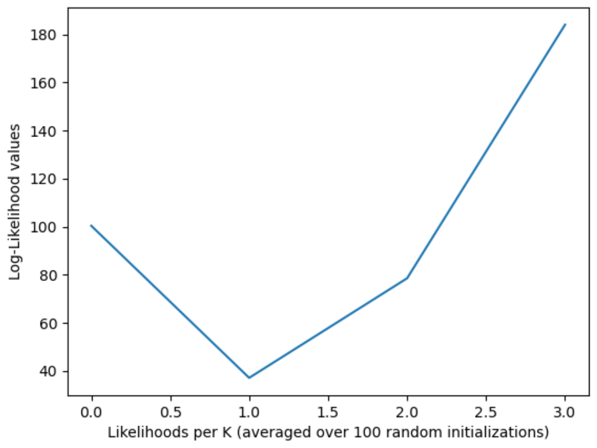

Figure 1: EM for Bernoulli Mixture Model.

(ii): Assume that the same data was infact generated from a mixture of Gaussians with 4 mixtures. Implement the EM algorithm and plot the log-likelihood (averaged over 100 random initializations of the parameters) as a function of iterations. How does the plot compare with the plot from part (i)? Provide insights that you draw from this experiment.

Ans: If we assume that the given dataset has been generated from a mixture of Gaussians with 4 mixtures (which is obviously not the case in reality), we get the following plot (Ref: Fig:), if averaged over 100 random initializations for the loglikelihood. **(iii):** Run the K-means algorithm with K = 4 on the same data. Plot the objective of K-means as a function of iterations.

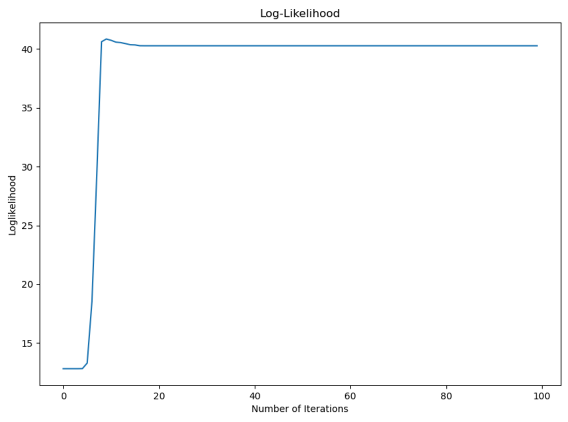

Figure 2: EM for Gaussian Mixture Model with K=4, averaged over 100 random initializations

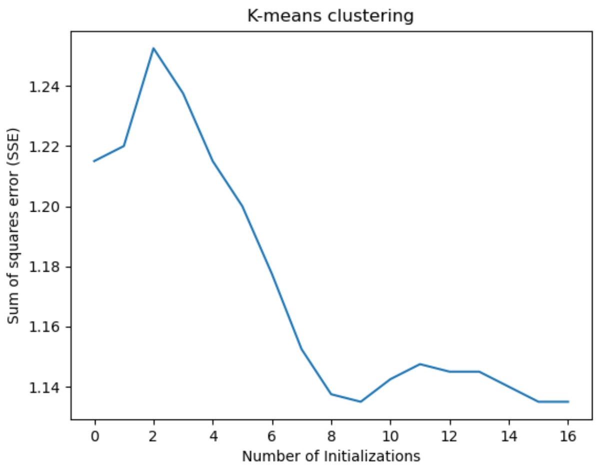

Figure 3: K-means Algorithm with K=4

Ans: The plot for K-Means with K=4 for the given dataset can be visualized via Fig: 3 which has been plotted by averaging over 100 random initializations.

(iv): Among the three different algorithms implemented above, which do you think you would choose to for this dataset and why?

Ans: Evidently from the plots provided as well as from my analysis of the dataset, it can be concluded that my previous assumption that the dataset has been generated from a Mixture of Bernoulli Distributions could infact be true. The Log-Likelihood for the implementation with Mixture of Bernoulli shows maximum value over 100 random initializations.

Question: 2

You are given a data-set in the file A2Q2Data train.csv with 10000 points in (R^{100}, R) (Each row corresponds to a datapoint where the first 100 components are features and the last component is the associated y value).

(i): Obtain the least squares solution $\mathbf{w}_{ML}$ to the regression problem using the analytical solution.

Ans: The analytical solution can be obtained via an algebraic approach of solving

$$\mathbf{w}_{ml} = (X^T X)^{-1} \cdot X^T \cdot Y$$

The Test Loss with $\mathbf{w}_{ml}$: `13.614832190845886` , Test Loss with $\mathbf{w}_r$: `11.002907809176868`

(ii): Code the gradient descent algorithm with suitable step size to solve the least squares algorithms and plot $\|\mathbf{w}^t - \mathbf{w}_{ML}\|_2$ as a function of t. What do you observe?

Ans: The required code has been attached with the Report in the submitted .zip file. The Plot of the observations can be visualized from Fig: 4. It has been observed that the value of the function decreases with respect to time steps. After certain number of iterations, an asymptotic behaviour can also be noticed

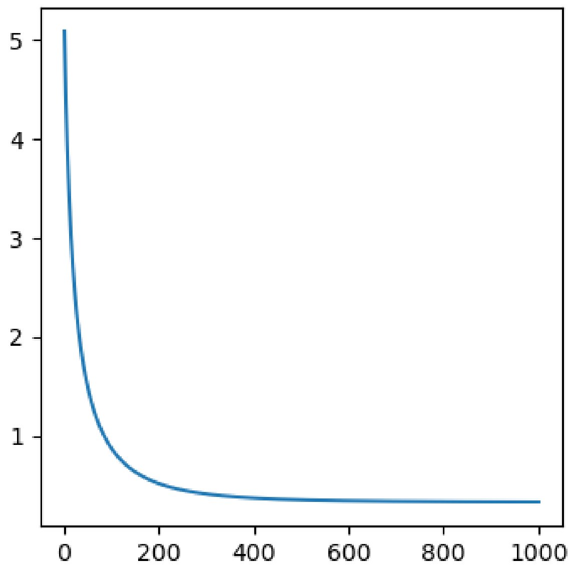

Figure 4: Gradient Descent

(iii): Code the stochastic gradient descent algorithm using batch size of 100 and plot $\|\mathbf{w}^t - \mathbf{w}_{ML}\|_2$ as a function of t. What are your observations?

Ans: Using a batch size of 100, the required code has been implemented and the following plot has been obtained. It is observed that the stochastic gradient descent approach converges faster than the gradient descent approach alone, but eventually both will converge.

(iv): Code the gradient descent algorithm for ridge regression. Cross-validate for various choices of λ and plot the error in the validation set as a function of λ. For the best λ chosen, obtain $\mathbf{w}_R$. Compare the test error (for the test data in the file 'A2Q2Data test.csv') of $\mathbf{w}_R$ with $\mathbf{w}_{ML}$. Which is better and why?

Ans: The gradient descent algorithm for ridge regression has been implemented and supplied in the .zip file along

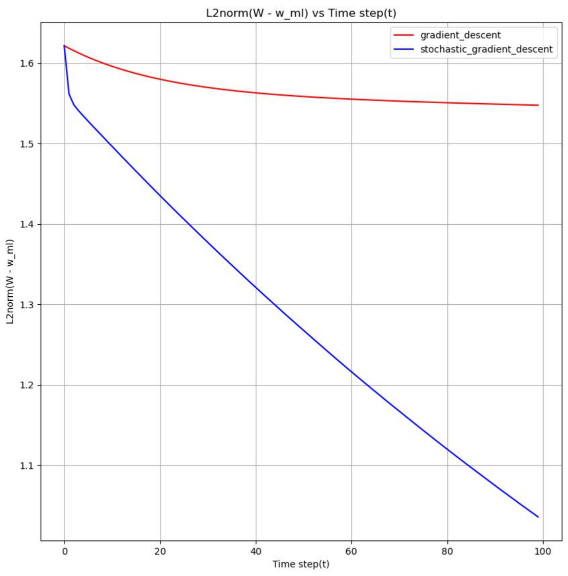

Figure 5: Stochastic Gradient Descent and Gradient Descent

with the submissions. The Test Loss with $\mathbf{w}_{ml}$: `13.614832190845886` , Test Loss with $\mathbf{w}_r$: `11.002907809176868` For the best lambda under the experimental circumstances, the test error is equal to `11.002907809176868` which is less than regression error obtained previously. Thus it can be concluded that ridge regression performs better than linear regression, all thanks to the regularization parameter.

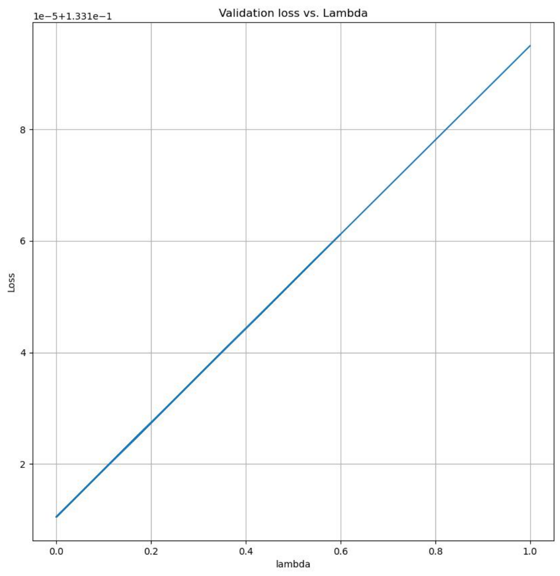

Figure 6: Validation Loss vs Lambda