Introduction to Machine Learning: A Review

Bahman Moraffah

June 2019
ASU-EECE-06-2019

School of Electrical, Computer, and Energy Engineering
Arizona State University
Tempe, AZ 85287

Abstract

In this report, we survey various methods in supervised and unsupervised learning. We study supervised methods such as regression and classification. We briefly discuss dimension reduction methods. We then investigate unsupervised parametric models such as K-means, Gaussian mixture models and generalize them to infinite dimension mixture model using Dirichlet process mixture model. We will also examine the Bayesian inference sampling. Also discussed is Monte Carlo and Markov Chain Monte Carlo (MCMC) techniques which serve as the backbone for estimation techniques. In particular, we study Gibbs sampler apply it to Dirichlet process mixture model in order to sample from the posterior. Lastly, a worked out example for each topic is discussed and the results are analyzed.

Contents

List of Figures

1 Introduction

With advancement in data science and high dimensional data, the need to learn machine learning methods increase. In this report, we present an overview of most used and common machine learning algorithms in addition to simple examples that analyze them. This report contains both supervised and unsupervised methods. We explore parametric linear regression and classification models and briefly generalize them to nonparametric regression/classification. Unsupervised parametric models such as K-means and Gaussian mixture model (GMM) are reviewed and simple examples are analyzed. We study Bayesian inference methods that are commonly used to do inference in Bayesian learning setups. We briefly survey expectation-minimization algorithm to maximize the log-likelihood. Moreover, Markov chain Monte Carlo (MCMC) methods are discussed. In particular, we study Gibbs sampler to draw from distributions which are generally difficult to compute. A statistical review for variational Bayes based on Kullback-Leibler divergence is provided.

2 Supervised Learning

2.1 Linear Regression

Assume the data $\mathbb{D} = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ where $X_j \in \mathbb{R}^d$ and $Y_j \in \mathbb{R}$ are observed. We would like to predict the outcome of Y given a new X. We define the conditional prediction risk to be

$$R(\hat{f}) = \mathbb{E}[(Y - \hat{f}(X))^2 | \mathbb{D}] \tag{1}$$

where $\hat{f}$ is the regression function. In addition, we define the prediction risk to be

$$r(\hat{f}) = \mathbb{E}_{\mathbb{D}}[R(\hat{f})]. \tag{2}$$

It can be shown the true regression function is $f(x) = \mathbb{E}[Y|X = x]$. A linear predictor is a function $g(x) = \beta^T x$ and the best linear predictor is the g that minimizes the prediction risk; in other words, the best linear predictor is the β that minimizes the prediction error.

Lemma 1 *Assume that* $\hat{\Sigma} = \frac{1}{n}\sum_j X_j X_j^T$ *is non-singular, then* β *that minimizes the empirical prediction risk, training error,* $\hat{r}(\beta) = \frac{1}{n}\sum_j (Y_j - \beta^T X_j)^2$*, is*

$$\hat{\beta} = \hat{\Sigma}^{-1}\hat{\alpha} \tag{3}$$

where $\hat{\alpha} = \frac{1}{n}\sum_j Y_j X_j$.

It is shown that for low dimensional linear regression the estimator $\hat{\beta}$ is very close to the actual minimizer that minimizes the prediction risk.

Theorem 1 *Assume* $\beta*$ *is the prediction risk minimizer. If the support of all joint distributions over* (X, Y) *are compact, then*

$$r(\hat{\beta}) - r(\beta^*) = O_P(\sqrt{\frac{1}{n}}) \tag{4}$$

where n *is the number of observed data*[1].

2.2 High Dimensional Linear Regression

If $d > n$ then $\hat{\Sigma}$ is singular and we no longer may use the least square error approach. There are various ways to solve the linear regression problem in high dimension including:

- Lasso
- Ridge regression
- Principal component analysis
- Forward stepwise regression.

In this report we only talk about Lasso and PCA. In sections 2.2.1, 2.2.2 we study some properties of lasso and ridge regression, respectively. Also in section 2.4 we discuss principal component analysis in detail.

[1] X_n is $O_{\mathrm{P}}(a_n)$ if $\mathrm{P}\{|\frac{X_n}{a_n}| > M\} \leq \epsilon$.

2.2.1 Lasso

To solve the linear regression problem, we look for an appropriate k-dimensional subset of Covariates such that $\Sigma_{|k}$ is invertible. Assume $\mathcal{K}$ is the subset of dimension k and $X_{\mathbb{K}}$ is the collection of Covariates such that Covariates belong to $\mathbb{K} \in \mathcal{K}$. We would like to choose $\mathbb{K}$ such that

$$\mathbb{E}[(Y - \beta_{\mathbb{K}}^T X_{\mathbb{K}})^2] \tag{5}$$

is minimized. This optimization problem is equivalent to

$$\begin{aligned} &\mathbb{E}[(Y - \beta^T X)^2] \\ &\text{subject to } ||\beta||_0 \leq k \end{aligned} \tag{6}$$

where $||\beta||_0$ is the number of non-zero elements, norm zero. Unfortunately, this problem is not a convex problem and the minimization is NP-hard. We therefore approximate Equation 6 with a convex function. This approach lead to "Lasso". The best way to relax this non-convex problem is replace norm zero by its convex approximation of it, meaning $||\cdot||_1$. Relaxing this condition gives us the following convex problem that is known as lasso. The lasso estimator $\hat{\beta}$ is defined as the minimizer of

$$\sum_j (Y_j - \beta^T X)^2 + \lambda ||\beta||_1 \tag{7}$$

for hyperparameter λ.

It is easy to check this problem is convex and the estimator can be found efficiently. This estimator is sparse for large values of λ. There are theoretical results that show the lasso estimator is close enough to the actual estimator under some conditions [1, 2].

2.2.2 Ridge Regression

Another way to relax the problem in 6 is to replace the $||\cdot||_0$ by $||\cdot||_2$. Hence, the problem constitutes to find the β-minimizer for the following problem:

$$\sum_j (Y_j - \beta^T X)^2 + \lambda ||\beta||_2 \tag{8}$$

for hyperparameter $\lambda \geq 0$. It can be shown that the minimizer is

$$\hat{\beta} = (\hat{\Sigma} + \lambda I)^{-1} \hat{\alpha} \tag{9}$$

for $\hat{\Sigma}$ and $\hat{\alpha}$ defined in Lemma 3. It is proven [3] that risk of the estimator, under mild conditions, with high probability is close to the risk of actual estimator.

2.3 Linear Classification

In this report, we only consider binary classification problem. Generalizing classification to K class may be done similarly. For the classification problem, we observe $\mathbb{D} = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ where $X_j \in \mathbb{R}^d$ and $Y_j \in \{0.1\}$. The goal is to predict the label Y upon receiving new feature X. A classification rule, i.e. classifier, is a hypothesis $h : \mathcal{X} \subset \mathbb{R}^d \to \{0, 1\}$ that predicts Y from received

X and outputs a decision region. Linear classifiers output boundaries that are linear. This region can be found through

$$h(x) = I(\beta_0 + \beta^T x > 0) \tag{10}$$

for indicator function I and coefficients β. The decision boundary is thus defined as $\beta_0 + \beta^T x = 0$. To find coefficients β, we need to minimize the risk. The classification risk is defined as

$$R(h) = \mathbb{P}(Y \neq h(X)). \tag{11}$$

Lemma 2 *The decision rule the minimizes the classification risk is*

$$h^*(x) = \begin{cases} 1 & \text{if } f(x) > 1/2 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

where $f(x) = \mathbb{P}(Y = 1|X = x)$.

However, we cannot compute the minimizer h this way since the distribution is not known. To do so, we can estimate the classifier by replacing $f(x)$ but its plug-in estimator $\hat{f}(x)$ in Equation 12. It can be shown [4, 2] that risk of plug-in estimator is as large as the square root of plug-in estimation error.

2.3.1 Empirical Risk Minimization

Another way to approach this problem is to minimize the empirical risk over all β. We define $\hat{\beta}$ to be the minimizer of the empirical risk given as

$$\hat{R}(h) = \frac{1}{n} \sum_{j=1}^{n} I(Y_j \neq h(X_j)). \tag{13}$$

It is shown that if $\hat{h}$ is the empirical risk minimizer, then $R(\hat{h}) - R(h^*) = O_P(\sqrt{\frac{(d+1)\log n}{n}})$.

2.3.2 Logistic Regression

Logistic regression is a discriminative approach to the binary classification problem and to estimate $f(x)$. In this model we define

$$f(x) = \mathbb{P}(Y|X = x) = \frac{\exp(\beta_0 + \beta^T x)}{1 + \exp(\beta_0 + \beta^T x)} \tag{14}$$

and by maximizing the likelihood using E-M algorithm, we can find β_0 and β.

2.3.3 Gaussian Discriminant Analysis

Suppose we have the following model:

$$\begin{aligned} &Y|\pi \sim \text{Bernoulli}(\pi) \\ &X|Y = 0 \sim \mathcal{N}(\mu_0, \Sigma_0) \\ &X|Y = 1 \sim \mathcal{N}(\mu_1, \Sigma_1). \end{aligned} \tag{15}$$

Algorithm 1 Principal Components Analysis (PCA)

procedure PCA(X)
  Compute $\hat{\Sigma} = \frac{1}{n}\sum(X_i - \bar{X})(X_i - \bar{X})^T$
  Compute eigenvalues and order them $\lambda_1 \geq \lambda_2 \cdots \geq \lambda_d$ of $\hat{\Sigma}$
  Compute corresponding eigenvectors $v_1, \ldots, v_d$ of $\hat{\Sigma}$
  Pick the best k-dimensional linear subspace
  Define the reduced data to be $\Pi_k(X) = \bar{X} + \sum_{j=1}^{k} \alpha_j v_j$ where $\alpha_j = \langle X, v_j \rangle$.

The log-likelihood of the data is given by:

$$\ell(\mu_0, \mu_1, \Sigma_1, \Sigma_2) = \log \prod_{j=1}^{n} p(X_j, Y_j | \pi, \mu_0, \mu_1, \Sigma_1, \Sigma_2) \tag{16}$$

which can be expanded in terms of Equation 15. By maximizing the log-likelihood, we can find $\pi, \mu_0, \mu_1, \Sigma_1$, and Σ_2.

2.4 Principle Component Analysis

In high dimensional data analysis, we need to approximate the lower dimension so that we do other tasks in the lower dimensional space. Principle component analysis (PCA) algorithm assumes that the number of observations n is a lot less than the number of features, $d \gg n$. The goal is to find the best $k \ll d$ dimensional linear subspace to approximate the space. Without loss os generality, we can assume that the mean is zero and we would like to find the projection of X onto the k-dimension space s_k. Let $\Sigma = \mathbb{E}(XX^T)$ be the Covariance matrix. Assume $v_1, v_2, \ldots, v_d$ be the eigenvectors corresponding to the ordered eigenvalues $\lambda_1 \geq \lambda_2 \cdots \geq \lambda_d$. One can decompose the Covariance matrix as follows:

$$\Sigma = V \Lambda V^T$$

where $V = [v_1, \ldots, v_d]$ is the matrix containing eigenvectors and $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_d)$.

Theorem 2 *The best k-dimension linear subspace is the space spanned by $v_1, \ldots, v_k$, i.e.,*

$$\Pi_k(X) = \sum_{j=1}^{k} \alpha_j v_j \tag{17}$$

where $\alpha_j = \langle X, v_j \rangle$.

the risk function satisfies

$$R_k = \mathbb{E}\big(||X - \Pi_k(X)||^2\big) = \sum_{j=k+1}^{d} \lambda_j. \tag{18}$$

In practice, however, we replace the Covariance matrix by the sample Covariance matrix. The algorithm is summarized in Algorithm 1 [5, 1, 3].

Example: Consider a three dimensional data. We would like to find the best 1-D linear subspace. To do so, we draw 100 data points from a normal distribution $\mathcal{N}(0, \text{diag}(10, 5, 1))$ and we run the PCA algorithm. Figure 1 shows the best 1-D subspace derived using PCA algorithm 1.

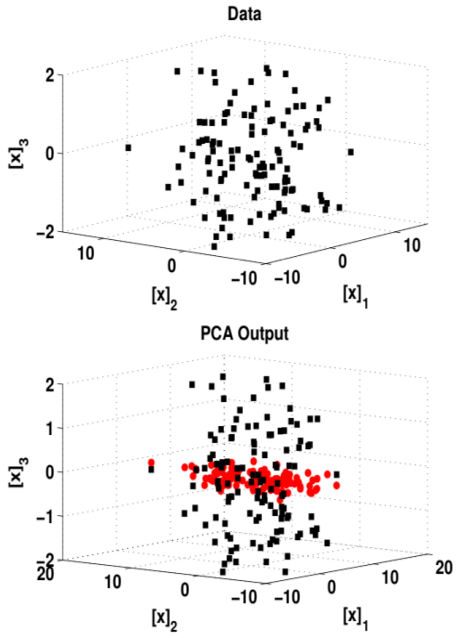

Figure 1: (a) Data (top) (b) Best 1-D linear subspace (bottom)

2.5 Factor Analysis

One of the dimension reduction methods is *factor analysis*. To this end, we assume that the number of observations n is a lot less than the number of features, $d \gg n$. The idea behind factor analytic approach is to use independency between features to reduce the dimension. It is worth mentioning that factor analysis is similar to PCA but not identical. Factor analysis is a generalization of PCA which is based on maximum-likelihood.

Assume $X = (X_1, \ldots, X_d) \in \mathbb{R}^d$ is a d-dimension vector with mean $\mu = \mu_1, \ldots, \mu_d$. We write each feature as a linear combinations of k factors plus noise, i.e., $X_j = \mu_j + \sum_{i=1}^{k} \gamma_{i,j} F_i + \epsilon_i$ where ϵ_i is the random noise with zero mean. We can rewrite this in terms of matrices:

$$X = \mu + \Gamma F + \epsilon$$

. There are many F's that satisfy this condition. However, we restrict ourselves to F with mean zero and variance I. We also assume F and ϵ are independent. Without loss of generality we can assume $\mu = 0$, then we have:

$$\begin{aligned} \text{Cov}(X) = \text{Cov}(\Gamma F + \epsilon) \implies & \text{Cov}(X) = \text{Cov}(\Gamma F) + \text{Cov}(\epsilon) = \Gamma \text{Cov}(F) \Gamma^T + \text{Cov}(\epsilon) \\ & \text{Cov}(X) = \Gamma \Gamma^T + \text{Cov}(\epsilon) \end{aligned} \tag{19}$$

One solution to this criterion satisfies $\Gamma = \Gamma Q$ and $F = Q^T F$, where Q is any orthogonal matrix [3, 1]. The best way to estimate F is through regression [3].

2.6 Independent Component Analysis (ICA)

To mathematically state this problem, assume we have signal $S \in \mathbb{R}^d$. However, we receive distorted version of S which is $X = AS$. Note that A is unknown and is called mixing matrix. Assume we repeat this n times and the goal is to recover signal S based on received X, meaning we look for W such that $S = WX$. To be able to define ICA, it is necessary to assume signals do not have Gaussian distribution but both X and S have zero mean. The idea is to find a metric to measure non-Gaussianity since finding the independent components is equivalent to finding the directions of largest non-Gaussianity. another way to study ICA is by minimizing the mutual information. Figure 2 compares PCA discussed in Section 2.4 to ICA introduced here.

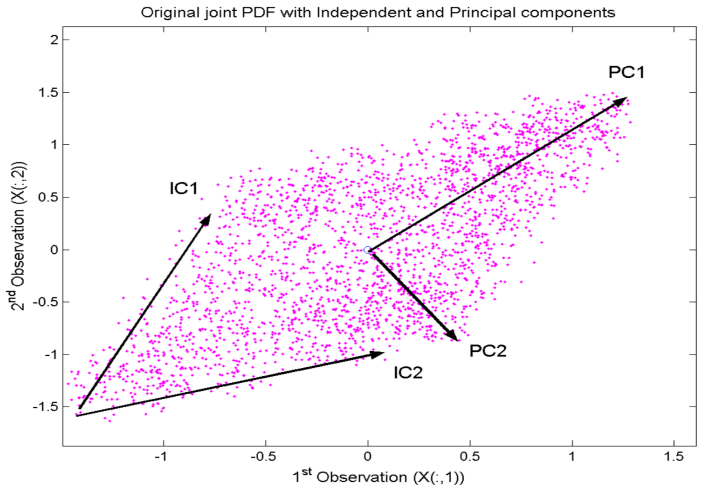

Figure 2: A comparison between ICA and PCA for two covariates

Note that maximizing the log-likelihood using stochastic gradient ascent learning, it turns out that we can update W using the following formula (calculations and details may be found in [6])

$$W = W + \lambda \begin{pmatrix} 1 - g(w_1^T X) \\ \dots \\ 1 - g(w_n^T X) \end{pmatrix} \tag{20}$$

where λ is the learning rate, $w_j \in \mathbb{R}^n$ is the j-th column of W,and $g(t) = \frac{1}{1+\exp(-t)}$.

3 Unsupervised Learning

3.1 K-means Clustering

One of the oldest unsupervised clustering methods is K-means clustering[1]. There are two school of thoughts on choosing K; some think of K as number of clusters, some interpret K as a tuning parameter that tends to be larger than the number fo actual clusters. In any case, we present K-means algorithm as a projection of data onto the set of cluster centers. To make this more precise, let $X_1, \ldots, X_n \sim \mathrm{P}$ where $X_j \in \mathbb{R}^d$ and cluster center collection $\mathcal{M} = \{\mu_1 \ldots . \mu_K\}$ where $\mu_j \in \mathbb{R}^d$. We choose $\mathcal{M}$ such that the following empirical risk is minimized:

$$\hat{R}(\mathcal{M}) = \frac{1}{n} \sum_{j=1}^{n} ||X_j - \Pi_{\mathcal{M}}(X_j)||^2 \tag{21}$$

where $\Pi_{\mathcal{M}}(X) = \underset{\mu_j \in \mathcal{M}}{\operatorname{argmin}} ||\mu_k - X||^2$ is the projection of X onto $\mathcal{M}$. Note that the algorithm that minimizes Equation 21 is K-Means clustering algorithm. We summarize this algorithm in 2.

Algorithm 2 K-Means Clustering

1: Form $\mathcal{M}$ b y choosing K cluster centers $\mu_1, \ldots \mu_K$
2: Define $c_j = \underset{i}{\operatorname{argmin}} ||\mu_i - X_j||$
3: i-th Cluster $C_i = \{X_j : c_j = i\}$
4: **Update the Cluster Center:**

$$c_j \leftarrow \frac{1}{n_j} \sum_{i: X_i \in C_j} X_i$$

where n_j is the number of data points in C_j
5: Repeat till convergence
6: **Return:** $\mathcal{M} = \{\mu_1 \ldots . \mu_K\}$

We Use K-means clusgtering algorithm to cluster 500 data points into two clusters $\{-1, +1\}$. As shown in Figure 3 the means for the two clusters converges to the sample means of the data. We also plot the risk function in Equation 21. Figure 4 demonstrates this risk function. Note that K-means++ algorithm is introduced to choose the initial values. We do not discuss this method here, however, one can find the detailed discussion on K-means++ in [7].

3.2 Gaussian Mixture Model

Mixture models are one of the most popular way to estimate the density of data as well as clustering them. Assume the parametric family $\{p(\cdot|\theta) : \theta \in \Theta\}$, define mixture model as

$$\begin{aligned} Z &= (z_1, \ldots, z_K) \sim \text{multi}(\pi) \\ X_j|\theta, Z &\sim p(\cdot|\theta_{z_j}) \qquad j = 1, 2, \ldots, N \end{aligned} \tag{22}$$

where parameters $\pi = (\pi_1, \ldots, \pi_K)$ and $\{\theta_j\}_{j=1}^K$ are not known. If the set of parametric family is Gaussian, we call this mixture model, Gaussian mixture model (GMM) and write:

$$p(x) = \sum_{j=1}^{K} \pi_j \mathcal{N}(x; \mu_{z_j}, \sigma_{z_j}) \tag{23}$$

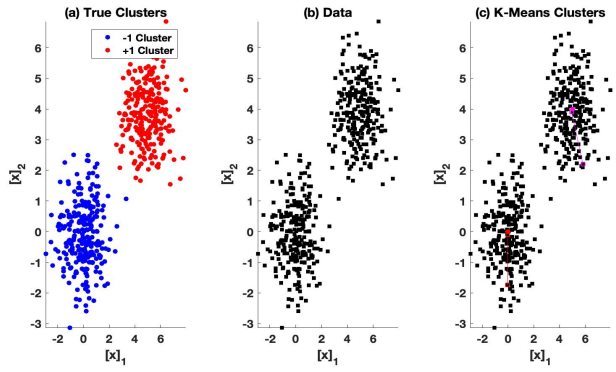

Figure 3: (a) Data points into two clusters (b) Data points with no label (c) K-means clustering algorithm applied to data to cluster data into $K = 2$

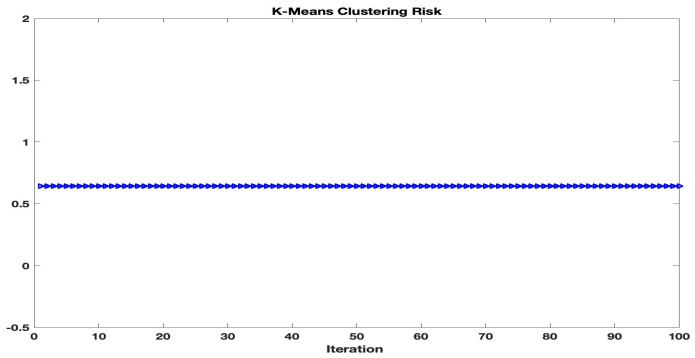

Figure 4: Risk function of K-means clustering for $K = 2$

For example, Figure 5 depicts mixture of two Gaussians. for this picture $\pi = (0.25, 0.75)$, $\mu_1 = -1.5, \mu_2 = 2.5$, and $\sigma_1 = \sigma_2 = 1$. As seen in Equation 22, GMM may be used for clustering. We use this clustering method for $N = 150$ data points drawn from Gaussian distributions. Figure 6 shows the actual cluster and the estimated distribution using the mixture of two Gaussians.

3.2.1 Maximum Likelihood Estimation for GMM

As mentioned in Section 3.2, parameters $\pi = (\pi_1, \ldots, \pi_K)$ and $\{\theta_j\}_{j=1}^K$ are not known. To estimate these parameters we use maximum likelihood estimator with respect to unknown parameters based on i.i.d. observations $X_1, \ldots, X_N \sim \mathrm{P}$. The likelihood function is

$$\mathcal{L} = \prod_{j=1}^{N} \Big(\sum_{k=1}^{K} \pi_j \mathcal{N}(x; \mu_k, \sigma_k) \Big) \tag{24}$$

To find the parameter estimators we maximize the log-likelihood instead, meaning

$$\ell = \sum_{j=1}^{N} \log \Big(\sum_{k=1}^{K} \pi_j \mathcal{N}(x; \mu_k, \sigma_k) \Big) \tag{25}$$

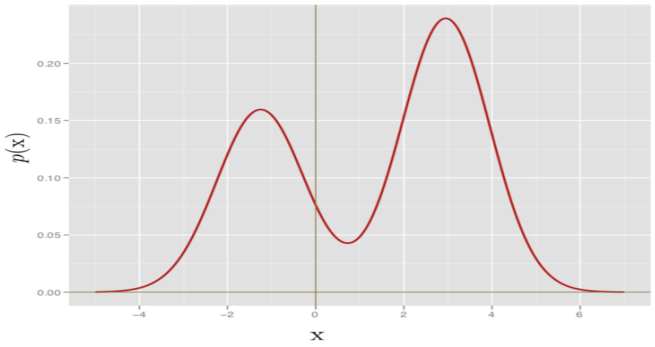

Figure 5: Mixture of Gaussians for $K = 2$

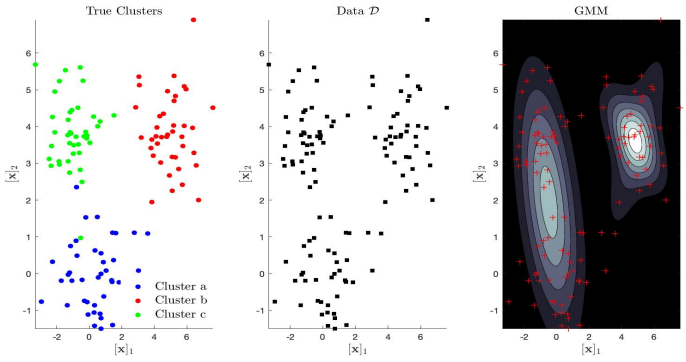

Figure 6: Mixture of Gaussians for $K = 2$ even though there are three clusters

however, Equation 25 is not jointly convex and hence it is not clear what method is the best to optimize this. One convenient and popular way to optimize this is through using E-M Algorithm that will be discussed in Section 4.1.

3.3 Infinite Mixture Model

Section 3.2 investigates the mixture models, however there is an important question to answer. How can one choose K? There are various methods to choose K such as cross validation. Nevertheless, to be able to use the discussed methods, we need to specify K which can be a difficult task. The main idea behind infinite mixture model is that we let K go to infinity so that we can create new clusters as needed. We make this idea more precise in the next two sections.

3.3.1 Dirichlet Process

The most popular nonparametric model is Dirichlet process invented by Ferguson [8]. To estimate the distribution through a Bayesian perspective we put a prior G on the space of infinite dimension, i.e.,

$$\begin{aligned} F|G &\sim G \\ X_j|F &\sim F \qquad j = 1, \dots, N \end{aligned} \tag{26}$$

The most popular Bayesian nonparametric prior is Dirichlet process [9]. Dirichlet process introduces a probability random measure over the space of distributions. Dirichlet process with concentration parameter α and base measure H is shown with $\mathrm{DP}(\alpha,\ H)$. To define Dirichlet process, assume that $A_1, \ldots, A_k$ is a partition of the space Θ, we say G is a Dirichlet process if the vector $(G(A_1), \ldots, G(A_k)) \sim \mathrm{Dir}(\alpha H(A_1), \ldots, \alpha H(A_k))$, where Dir is Dirichlet distribution. Although this definition completely specify Dirichlet process, this definition is not constructive. Sethuraman introduces a constructive method for Dirichlet process [10]. This method is based on stick-breaking process. A draw from Dirichlet process $G|H \sim \mathrm{DP}(\alpha, H)$ can be constructed as

$$\begin{aligned} \pi &\sim \mathrm{GEM}(\alpha) \\ \theta_j|H &\sim H \qquad j = 1, 2, \ldots \\ G &= \sum_{j=1}^{\infty} \pi_j \delta_{\theta_j} \end{aligned} \tag{27}$$

Where $\delta_\theta(\Theta) = 1$ if $\theta \in \Theta$ and $\delta_\theta(\Theta) = 0$ if $\theta \notin \Theta$. Note that $\mathrm{GEM}(\alpha)$ distribution is given by

$$\begin{aligned} \pi'_j &\sim \mathrm{Beta}(1, \alpha) \\ \pi_j &= \pi'_j \prod_{i=1}^{j-1} (1 - \pi'_i). \end{aligned} \tag{28}$$

This construction of Dirichlet process is known as stick-breaking process [11]. The Dirichlet process for $\mathcal{N}(0, 1)$ and for 10000 MCMC sample is depicted in Figure 7.

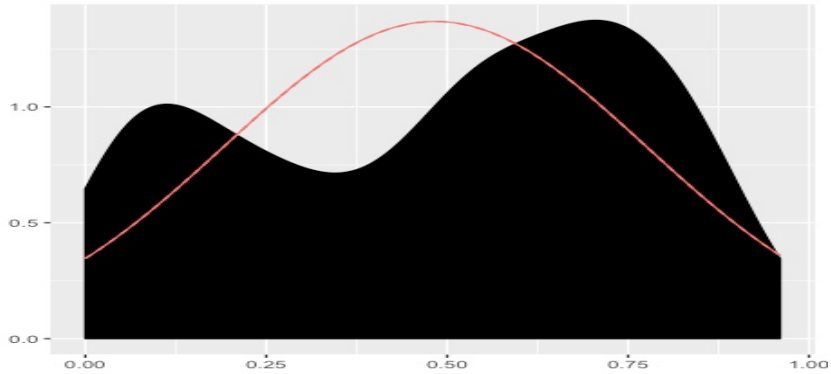

Figure 7: A Draw from Dirichlet Process

It can easily be shown that $\mathbb{E}[G] = H$. It can also be shown the posterior distribution of G upon receiving measurements $X_1, \ldots X_N \sim G$ is also a Dirichlet process. The following theorem demonstrates this observation.

Theorem 3 *Assume$X_1, \ldots X_N \sim G$ and G has the prior of Dirichlet process on it. Then, the posterior distribution is a Dirichlet process, i.e.,*

$$G|X_1, \ldots, X_N \sim DP(\alpha + n, H + \sum_{j=1}^{N} \delta_{X_j}) \tag{29}$$

Since the posterior is again a Dirichlet process with updated parameters, we can sample from it as we do for the prior.

3.3.2 Dirichlet Process Mixture Model

Dirichlet process is exploited as a prior over distribution space and can estimate the distributions. However, we sometimes need to estimate the density of a continuous random variable. Dirichlet process is not appropriate prior over the space of densities since (a) Dirichlet process is discrete with probability one (b) Dirichlet process does not have density. This leads us to generalize Dirichlet process to Dirichlet process mixture model. Assume $X_1, \ldots, X_N \sim p$, infinite mixture model can be summarized as

$$\begin{aligned} G|H &\sim \mathrm{DP}(\alpha, H) \\ \theta_j|G &\sim G \\ X_j|\theta_j &\sim p(\cdot|\theta_j). \end{aligned} \tag{30}$$

One can marginalize out G and rewrite the Dirichlet process mixture model as:

$$\begin{aligned} \pi &\sim GEM(\alpha) \\ \theta_j|H &\sim H \\ z_j|\pi &\sim \mathrm{Cat}(\pi) \\ X_j|\theta_j, z_j &\sim p(\cdot|\theta_{z_j}). \end{aligned} \tag{31}$$

The Equation 31 can then be written as $p(x) = \sum_{j=1}^{\infty} \pi_j p(x|\theta_j)$, which can be seen that it is similar to 22 when $K \to \infty$. There is more technicality for this observation which can be found in [12]. We use this infinite mixture model using Gaussian $p = \mathcal{N}(\theta)$ and $\mathcal{NIW}$ distribution as base distribution. Figure 8 shows the Dirichlet process mixture model.

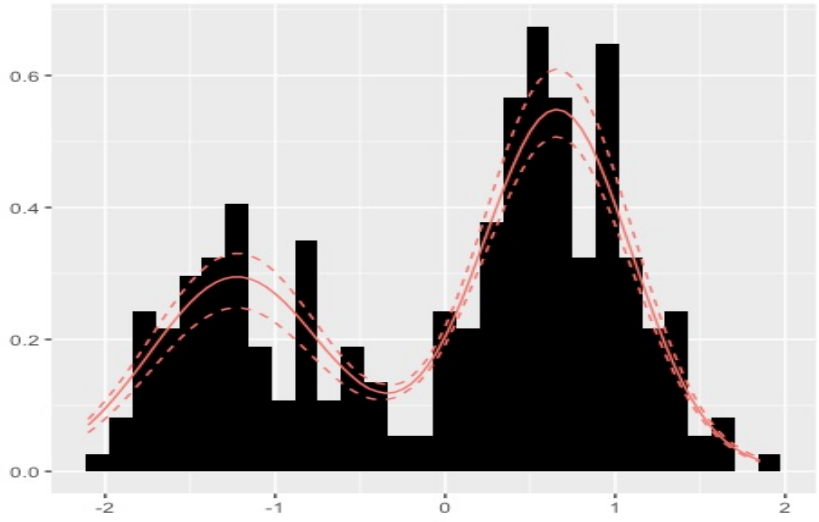

Figure 8: Dirichlet process mixture model for Gaussian distribution

Dirichlet process and Dirichlet process mixture models may not address the dependency among samples. Dependent Dirichlet process introduces a nonparametric prior over evolving mixture models. The dependent Dirichlet process (DDP) originally introduced by MacEachern led to the

development of the DDP mixture model [13, 14]. The DDP mixture model generalizes Dirichlet process mixture model by considering birth, death and transition processes for the clusters in the model and has variety of applications including in tracking multiple objects [15, 16, 17]. A special case of DDP is hierarchical Dirichlet process that provides a relationship between grouped data [18, 19, 20, 21, 22]. A generalization of Dirichlet process is two-parameter Poisson-Dirichlet process that was introduced by Ptiman and Yor [23]. This process has found many applications in topic modeling, word clustering, and tracking [22, 24, 25, 26].

4 Inferential Methods

4.1 Expectation-Maximization Algorithm

Expectation-Maximization Algorithm (EM Algorithm) is an algorithm to alternatively maximize the likelihood function. In particular, given data points $X_1, \ldots, X_N \sim p(x|\theta)$, we would like to estimate unknown parameters θ. We have latent variables model $p(x, z|\theta)$ with z being the latent variable (which is assumed to take finite number of values). Therefore, $p(x|\theta) = \sum_z p(x, z|\theta)$. To estimate the parameters we need to maximize the likelihood $\mathcal{L}(\theta)$. The goal is to find the maximum likelihood estimator $\hat{\theta}_{\text{MLE}} = \underset{\theta}{\text{argmin}}\mathcal{L}(\theta)$. Instead, we can optimize the log-likelihood

$$\ell(\theta) = \sum_x \log \sum_z p(x, z|\theta). \tag{32}$$

However, maximizing the likelihood is troublesome since the summation make the distribution a multimodal distribution. We instead find a lower bound and maximize the lower bound to be the closest to the log-likelihood, meaning for distribution over all z, Q

$$\log(p(x|\theta)) = \log(\sum_z p(x, z|\theta)) = \log(\sum_z Q(z)\frac{p(x, z|\theta)}{Q(z)}) \geq \sum_z Q(z) \log \frac{p(x, z|\theta)}{Q(z)} \tag{33}$$

where the last inequality is due to Jensen's inequality. For this bound to be tight, $Q(z) \propto p(x, z|\theta)$. Furthermore, $Q(z) = p(z|x, \theta)$ which is the posterior distribution of Z given the data. The inequality in Equation 33 is called evidence lower bound (ELBO) and show by $\text{ELBO}_Q(\theta)$. We can rewrite Equation 33 as follows:

$$\log(p(x|\theta)) \geq \text{ELBO}_Q(\theta). \tag{34}$$

Intuitively speaking, EM algorithm alternates between updating Q and θ by first setting $Q(z) = p(z|x, \theta)$ and hence $\log(p(x|\theta)) = \text{ELBO}_Q(\theta)$ and then by maximizing $\text{ELBO}_Q(\theta)$ with respect to θ for fixed Q. EM algorithm is summarized in Algorithm 3.

Algorithm 3 EM Algorithm

1: **Initialize** $\theta_0 \in \Theta$
2: **for** **do**$t = 0, 1, 2, \ldots$
3: **E-Step** $\mathcal{Q}(\theta, \theta_t) = \mathbb{E}_{\theta_t}[\log(p(x, z|\theta))|X = x]$
4: **M-Step** $\hat{\theta}_{t+1} = \underset{\theta}{\text{argmin}}\mathcal{Q}(\theta, \theta_t)$
5: Iterate till convergence

4.2 Markov Chain Monte Carlo Methods

Markov chain Monte Carlo (MCMC) methods are the most used inferential methods which provide exact samples from the target distribution for any problem with probabilistic interpretation with some parameters e.g., many problems in machine learning, optimization, and statistics. These inferential methods utilize independent samples of distribution to analyze the distribution for which explicit computation of the distribution is difficult.

Many inference tasks such as computing the marginal can be represented as the integral and therefore, as the expected value of some appropriately chosen function [27, 5]. Hence, due to the law of large number, it can be described by the empirical mean of independent random variables.

There are various Monte Carlo methods that offer different approaches to generate independent samples; most of which are based on the random walk. These methods are based on choosing a proposal distribution and thus are very sensitive to the step size. The idea to generate independent samples is to design a first order Markov chain with the target stationary probability distribution where in the limit, distribution of the samples converges to the target distribution. It is shown that due to the ergodic theorem, the stationary distribution is approximated by the empirical measures of the random states of the MCMC sampler [28]. The main idea is originated from the simple observation that is stated in the following theorem.

Theorem 4 *the following statements are equivalent:*

(i) $\mathbf{X} \sim p(\mathbf{x})$

(ii) $(\mathbf{X}, \mathbf{U}) \sim \mathcal{U}nif\{(\mathbf{x}, \mathbf{u}) : 0 \leq \mathbf{u} \leq p(\mathbf{x})\}$.

Two of the main Markov chain Monte Carlo methods are Metropolis-Hasting and Gibbs sampling. In addition to random walk based MCMC methods, we explore the elegant slice sampling method to solve the issues with these methods. We briefly study some of the methods that are used in this thesis.

4.3 Generalized Importance Sampling

Importance sampling approach is an MCMC sampling method to estimate the expected value. This method is based on a proposal distribution and thus, relies on importance functions. Suppose $p(\mathbf{x}) = \bar{p}(\mathbf{x})/Z$ can be evaluated up to the normalizing constant Z. Choose proposal distribution $q(\mathbf{x})$ such that $q(x)$ is absolutely continuous with respect to $p(\mathbf{x})$ meaning $\text{supp}(p(\mathbf{x})) \subset \text{supp}(q(\mathbf{x}))$ and draw $\mathbf{x}_1, \ldots, \mathbf{x}_N \sim q(\mathbf{x})$ then, for some function h

$$\frac{1}{N} \sum_{j=1}^{N} \omega_j h(\mathbf{x}_j) \to \mathbb{E}_p[h(\mathbf{X})] = \int h(\mathbf{x}) q(\mathbf{x}) \frac{p(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x} \tag{35}$$

as $N \to \infty$, where $\omega_j = \frac{\tilde{\omega}_j}{\sum_j \tilde{\omega}_j}$ and $\tilde{\omega}_j = \frac{\bar{p}(\mathbf{x}_j)}{q(\mathbf{x}_j)}$. Therefore, the expected values is estimated using the importance functions $\{\tilde{\omega}_j\}_{j=1}^{N}$. It is shown this estimation is asymptotically consistent [29].

In this section, we provide a general framework for importance sampling based on dependent proposal distributions and adaptive algorithms where it provides an unbiased estimator for the target expected value. It is proven that dependency in the samples still preserves the unbiasedness property [30]. The following lemma shows that the modification of importance weights by a kernel preserves the unbiasedness of the estimator.

Lemma 3 *if p and q are distributions such that $p \ll q$ and importance weight $\tilde{\omega} = \frac{p(\mathbf{x})}{q(\mathbf{x})}$, then for any kernel $\mathbb{K}(\mathbf{x}, \mathbf{x}')$ with stationary distribution p*

$$\int \tilde{\omega} \mathbb{K}(\mathbf{x}, \mathbf{x}') q(\mathbf{x}) = p(\mathbf{x}') \tag{36}$$

Algorithm 4 Dynamic Importance Sampling

Input: $(\mathbf{x}, \tilde{\omega})$, and $\mathbb{K}(\mathbf{x}, \mathbf{x}')$ where $\tilde{\omega} = \frac{\bar{p}(\mathbf{x})}{q(\mathbf{x})}$
for k = 1,2, … **do**
Draw $\mathbf{x}' \sim \mathbb{K}(\mathbf{x}_k, \mathbf{x}')$
Compute $\gamma_k = \tilde{\omega} \frac{p(\mathbf{x}')\mathbb{K}(\mathbf{x}', \mathbf{x}_k)}{p(\mathbf{x})\mathbb{K}(\mathbf{x}_k, \mathbf{x}')}$
Draw $\mathbf{u} \sim \mathcal{U}nif(0, 1)$
$(\mathbf{x}_{k+1}, \tilde{\omega}_{k+1}) \leftarrow (\mathbf{x}', \frac{(1+\delta)\gamma_k}{c})$ if $u < c$
$(\mathbf{x}_{k+1}, \tilde{\omega}_{k+1}) \leftarrow (\mathbf{x}_k, \frac{(1+\delta)\tilde{\omega}_k}{1-c})$ if $u > c$
where $c = \frac{\gamma_k}{\gamma_k + \eta(\mathbf{x}_k, \omega_k)}$, $\delta > 0$ and η are either constant or independent

Since the kernel $\mathbb{K}(\mathbf{x}, \mathbf{x}')$ corresponds to the target distribution, it can correct the poor choice of proposal distribution. A dynamic approach to importance sampling is introduced in [31]. The intuition for this algorithm is that We summarize this method in Algorithm 1.

4.4 Metropolis-Hastings Algorithm

Metropolis-Hastings algorithm is the universal MCMC algorithm where it produces an ergodic Markov chain whose stationary distribution is the target distribution p. In particular, we aim to draw samples from the posterior distribution $p(\boldsymbol{\theta}|\mathbf{z})$. We draw samples $\boldsymbol{\theta}_k$ sequentially based on a Markov chain. We construct a Markov chain $\mathbb{K}$ such that for large enough k, $\boldsymbol{\theta}_k$ is drawn from the desired posterior distribution, i.e., $\mathbb{K}^k \to p(\boldsymbol{\theta}|\mathbf{z})$. Suppose that Markov chain $\mathbb{K}$ is irreducible[2] and aperiodic[3] whose stationary distribution is p. Since p is the stationary distribution, it follows the detailed balance condition,

$$\mathbb{K}(\mathbf{x}, \mathbf{y})p(\mathbf{y}) = \mathbb{K}(\mathbf{y}, \mathbf{x})p(\mathbf{x}) \tag{37}$$

Metropolis-Hastings algorithm starts by selecting an easy to implement conditional distribution $q(\cdot|\cdot)$ which is absolutely continuous with respect to the target distribution. Without loss of generality we can assume $q(\mathbf{x}|\mathbf{y})p(\mathbf{x}) > q(\mathbf{y}|\mathbf{x})p(\mathbf{y})$, and hence there exist an acceptance probability $0 \leq \alpha(\mathbf{x}, \mathbf{y}) \leq 1$ such that

$$q(\mathbf{x}|\mathbf{y})p(\mathbf{x}) = \alpha(\mathbf{x}, \mathbf{y})q(\mathbf{y}|\mathbf{x})p(\mathbf{y}) \implies \alpha(\mathbf{x}, \mathbf{y}) = \min\left\{1, \frac{q(\mathbf{x}|\mathbf{y})p(\mathbf{x})}{q(\mathbf{y}|\mathbf{x})p(\mathbf{y})}\right\} \tag{38}$$

It is shown in [32], the transition kernel associated with this equation follows

$$\mathbb{K}(\mathbf{x}, \Theta) = \int_{\Theta} \alpha(\mathbf{x}, \mathbf{y})q(\mathbf{y}|\mathbf{x})d\mathbf{y} + \mathbb{1}_{\mathbf{x}}(\Theta)\Big(1 - \int_{\Theta} \alpha(\mathbf{x}, \mathbf{y})q(\mathbf{y}|\mathbf{x})d\mathbf{y}\Big) \tag{39}$$

The Metropolis-Hastings algorithm associated with the target density p with conditional proposal distribution q produces a Markov chain $\{\mathbf{x}_k\}_k$ using 39. This method is referred to as Metropolis-Hastings algorithm and summarized in Algorithm 5.

Theorem 5 *Suppose that the Markov chain produced by Metropolis-Hastings is p-irreducible, then*

[2] All states can communicate with one another with positive probability in finite time.
[3] To ensure uniqueness of stationary distribution almost surely.

Algorithm 5 Metropolis-Hastings Algorithm.

Input: proposal distribution $q(\cdot|\cdot)$
Initialize $\mathbf{X}_0$ at random
for k=0,1,2,… **do**
  Draw $\mathbf{y}_{k+1} \sim q(\cdot|\mathbf{X}_k = \mathbf{x}_k)$
  Draw $\mathbf{u}_{k+1} \sim \mathcal{U}nif(0,1)$
  Compute the acceptance probability $\alpha(\mathbf{x}_k, \mathbf{Y}_{k+1})$ in 38
  if $\mathbf{u}_{k+1} \leq \alpha(\mathbf{x}_k, \mathbf{Y}_{k+1})$ **then**
    $\mathbf{x}_{k+1} \leftarrow \mathbf{y}_{k+1}$
  else
    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k$
Burn-in Dismiss the first $\mathbf{x}_1, \ldots \mathbf{x}_r$

a) For any function $g \in L_1(p)$,

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} g(\mathbf{x}_k) = \int g(\mathbf{x}) d\mathbb{P}(\mathbf{x}) \tag{40}$$

b) If $\mathbf{X}_k$ is aperiodic, then for every initial distribution ν

$$\lim_{N\to\infty} || \int \mathbb{K}^N(\mathbf{x}, \cdot)\nu(d\mathbf{x}) - p||_{TV} = 0. \tag{41}$$

Choice of q result in different Metropolis-Hastings algorithms. We study two main choices of q next.

Independent Metropolis-Hastings: When $q(\mathbf{x}|\mathbf{y})$ is independent of $\mathbf{Y}$, that is, $q(\mathbf{y}|\mathbf{x}) = q(\mathbf{y})$. This leads to an algorithm called Independent Metropolis-Hastings algorithm. In this case, the acceptance probability $\alpha(\mathbf{x}, \mathbf{y})$ is simplified to

$$\alpha(\mathbf{x}, \mathbf{y}) = \min\left\{1, \frac{q(\mathbf{x})p(\mathbf{x})}{q(\mathbf{y})p(\mathbf{y})}\right\}. \tag{42}$$

Although $\mathbf{Y}_k$'s are generated independently in Algorithm 5, the resulting samples $\mathbf{X}_k$'s are not i.i.d. since, for instance, probability of acceptance of $\mathbf{Y}_k$ relies directly on $\mathbf{X}_k$.

Random Walk Metropolis-Hastings: When q is symmetric, that is, $q(\mathbf{x}|\mathbf{y}) = q(\mathbf{y}|\mathbf{x})$. This leads to a method which is referred to as Metropolis-Hastings random walk algorithm. In this case proposal distribution q depends only on $|\mathbf{x} - \mathbf{y}|$. In this case, the acceptance probability $\alpha(\mathbf{x}, \mathbf{y})$ is given by

$$\alpha(\mathbf{x}, \mathbf{y}) = \min\left\{1, \frac{p(\mathbf{x})}{p(\mathbf{y})}\right\}. \tag{43}$$

Despite simplicity of computing the acceptance probability, this method tends to converge with slower rate. In addition, the random walk Metropolis-Hastings algorithm does not satisfy the uniform ergodicity property [33].

Algorithm 6 Two-Stage Gibbs Sampler to Sample form $p(\mathbf{x}, \mathbf{y})$

Initialize $(\mathbf{x}_0, \mathbf{y}_0)$
for k=1,2,... **do**
  $\mathbf{x}_k \sim p(\cdot|\mathbf{y}_{k-1})$
  $\mathbf{y}_k \sim p(\cdot|\mathbf{x}_k)$
Repeat until convergence

4.5 Gibbs Sampling

Gibbs sampling, also known as alternating conditional sampling, is a special case of Metropolis-Hastings algorithm where we partially update our vector. Gibbs sampler is mostly used when computing the conditional distribution is not complicated. The idea is to use the conditional distribution associated with the target distribution to generate samples from it. In the section, we first study the Gibbs sampler for two variables and then generalize it to a vector of random variables with straightforward conditional distributions. Gibbs sampling can be very slow if the parameters in target distribution are highly correlated. To avoid this issue, one can re-parametrize the parameters of interest. Consider the joint probability density $p(\mathbf{x}, \mathbf{y})$ on the product space $\mathcal{X} \times \mathcal{Y}$. Based on 4, define $\mathcal{E}(p) = \{(\mathbf{x}, \mathbf{y}, \mathbf{u}) : 0 \leq \mathbf{u} \leq p(\mathbf{x}, \mathbf{y})\}$. We generate

- $\mathbf{x}$ uniformly on $\mathcal{E}_{\mathbf{x}}(p) = \{\mathbf{x} : \mathbf{u} \leq p(\mathbf{x}, \mathbf{y})\}$ or equivalently from $\mathcal{E}_{\mathbf{x}}(p) = \{\mathbf{x} : \frac{\mathbf{u}}{p_{\mathbf{Y}}(\mathbf{y})} \leq p(\mathbf{x}|\mathbf{y})\}$.
- $\mathbf{y}$ uniformly on $\mathcal{E}_{\mathbf{y}}(p) = \{\mathbf{y} : \mathbf{u} \leq p(\mathbf{x}', \mathbf{y})\}$ or equivalently from $\mathcal{E}_{\mathbf{y}}(p) = \{\mathbf{y} : \frac{\mathbf{u}}{p_{\mathbf{X}}(\mathbf{x})} \leq p(\mathbf{y}|\mathbf{x}')\}$.
- $\mathbf{u}$ uniformly on $\{\mathbf{u} : 0 \leq \mathbf{u} \leq p(\mathbf{x}', \mathbf{y}')\}$.

However, if we leave $\mathbf{y}$ fixed and repeat this procedure infinite times, we end up with the samples from $p(\mathbf{x}|\mathbf{y})$. One can do the same along $\mathbf{y}$ and end up with the samples from $p(\mathbf{y}|\mathbf{x})$. We summarize this procedure in Algorithm 6. To illustrate the two-state Gibbs sampling procedure, we assume $\mathbb{X} = (\mathbf{X}, \mathbf{Y}) \sim \mathcal{N}(\mu, \Sigma)$ for unknown mean $\mu = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ and known covariance matrix

$$\Sigma = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}.$$

Assuming a uniform distribution as prior on μ, according to **??**, the conditional posterior distribution is given by

$$\begin{aligned} \boldsymbol{\theta}_1|\boldsymbol{\theta}_2, \mathbb{X} &\sim \mathcal{N}(\mathbf{X} + \rho(\boldsymbol{\theta}_2 - \mathbf{Y}), 1 - \rho^2) \\ \boldsymbol{\theta}_2|\boldsymbol{\theta}_1, \mathbb{X} &\sim \mathcal{N}(\mathbf{Y} + \rho(\boldsymbol{\theta}_1 - \mathbf{X}), 1 - \rho^2). \end{aligned} \tag{44}$$

9 demonstrates the Gibbs sampler for this model for 10,000 iterations and $\rho = 0.5$ and burn in $r = 100$ A Markov Chain Monte Carlo sampling method for Dirichlet processed in introduced in [34]. In particular, the Gibbs sampling implementation of Dirichlet process is discussed in [35, 36].

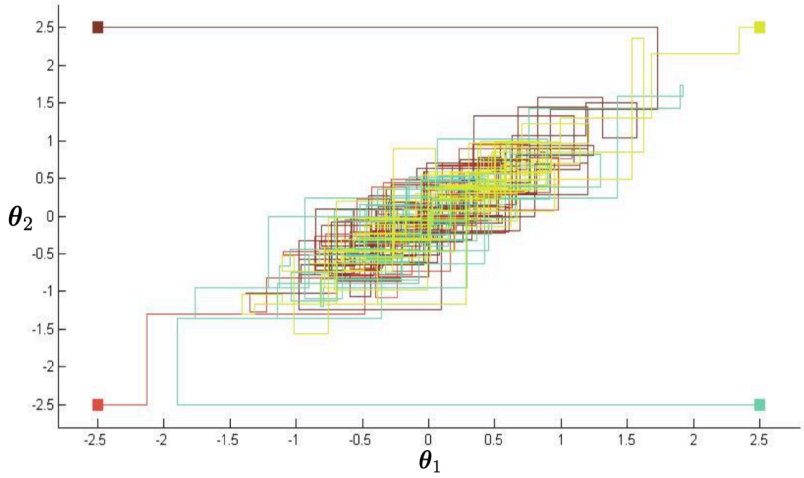

Figure 9: Gibbs sampler for a bivariate Gaussian distribution with 10,000 simulations.

5 Conclusion

In this report, we surveyed both supervised and unsupervised learning. We provided examples that support the theory. We discussed Bayesian inference methods and showed through examples that these methods converge to the posterior. Furthermore, we studied a Bayesian nonparametric model, Dirichlet process. We provided examples that demonstrated that Dirichlet process is discrete and thus is not appropriate to be used for density estimation. Instead, we use a generalization of Dirichlet process known as Dirichlet process mixture model to estimate the densities. This can be viewed as generalization of discussed Gaussian mixture model. Sampling methods are also provided.

References

[1] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Springer series in statistics New York, 2001.

[2] Larry Wasserman. *All of nonparametric statistics*. Springer Science & Business Media, 2006.

[3] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[4] Larry Wasserman. *All of statistics: a concise course in statistical inference*. Springer Science & Business Media, 2013.

[5] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[6] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent component analysis*, volume 46. John Wiley & Sons, 2004.

[7] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[8] Thomas S Ferguson. A Bayesian analysis of some nonparametric problems. *The annals of statistics*, pages 209–230, 1973.

[9] Yee Whye Teh. Dirichlet process. *Encyclopedia of machine learning*, pages 280–287, 2010.

[10] Jayaram Sethuraman. A constructive definition of Dirichlet priors. *Statistica sinica*, pages 639–650, 1994.

[11] Jayaram Sethuraman. A constructive definition of Dirichlet priors. *Statistica sinica*, pages 639–650, 1994.

[12] Nils Lid Hjort, Chris Holmes, Peter Müller, and Stephen G Walker. *Bayesian nonparametrics*, volume 28. Cambridge University Press, 2010.

[13] Steven N MacEachern. Dependent nonparametric processes. *Unpublished manuscript*, 1999.

[14] Steven N MacEachern. Dependent Dirichlet processes. *Technical rep. The Ohio State University*, 2000.

[15] Bahman Moraffah and Antonia Papandreou-Suppappola. Random infinite tree and dependent Poisson diffusion process for nonparametric Bayesian modeling in multiple object tracking. In *International Conference on Acoustics, Speech, and Signal Processing*, 2019.

[16] Bahman Moraffah. On the use of dependent nonparametric processes in multiple object tracking. Technical report, Arizona State University, June 2019.

[17] Bahman Moraffah and Antonia Papandreou-Suppappola. Dependent Dirichlet process modeling and identity learning for multiple object tracking. In *Asilomar Conference on Signals, Systems, and Computers*, 2018.

[18] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical Dirichlet processes. *Technical rep. UC Berkeley*, 2005.

[19] Yee W Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Sharing clusters among related groups: Hierarchical Dirichlet processes. In *Advances in neural information processing systems*, pages 1385–1392, 2005.

[20] Bahman Moraffah, Cesar Brito, Bindya Venkatesh, and Antonia Papandreou-Suppappola. Use of hierarchical Dirichlet processes to integrate dependent observations from multiple disparate sensors for tracking. In *22nd International Conference on Information Fusion*, 2019.

[21] Bahman Moraffah, Cesar Brito, Bindya Venkatesh, and Antonia Papandreou-Suppappola. Tracking multiple objects with multimodal dependent measurements: Bayesian nonparametric modeling. In *Asilomar Conference on Signals, Systems, and Computers*, 2019.

[22] Bahman Moraffah. Inference for multiple object tracking: A Bayesian nonparametric approach. *arXiv preprint arXiv:1909.06984*, 2019.

[23] Jim Pitman, Marc Yor, et al. The two-parameter poisson-dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, 25(2):855–900, 1997.

[24] Kar Wai Lim, Wray Buntine, Changyou Chen, and Lan Du. Nonparametric bayesian topic modelling with the hierarchical pitman–yor processes. *International Journal of Approximate Reasoning*, 78:172–191, 2016.

[25] Bahman Moraffah, Muralidhar Rangaswamy, and Antonia Papandreou-Suppappola. Nonparametric Bayesian methods and the dependent Pitman-Yor process for modeling evolution in multiple state priors. In *22nd International Conference on Information Fusion*, 2019.

[26] Issei Sato and Hiroshi Nakagawa. Topic models with power-law using pitman-yor process. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 673–682. ACM, 2010.

[27] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[28] Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

[29] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to MCMC for machine learning. *Machine learning*, 50(1-2):5–43, 2003.

[30] Steven N MacEachern, Merlise Clyde, and Jun S Liu. Sequential importance sampling for nonparametric Bayes models: The next generation. *Canadian Journal of Statistics*, 27(2):251–267, 1999.

[31] Faming Liang. Dynamically weighted importance sampling in Monte Carlo computation. *Journal of the American Statistical Association*, 97(459):807–821, 2002.

[32] Siddhartha Chib and Edward Greenberg. Understanding the Metropolis-Hastings algorithm. *The American statistician*, 49(4):327–335, 1995.

[33] Kerrie L Mengersen, Richard L Tweedie, et al. Rates of convergence of the Hastings and Metropolis algorithms. *The annals of Statistics*, 24(1):101–121, 1996.

[34] Radford M Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265, 2000.

[35] Hemant Ishwaran and Lancelot F James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96(453):161–173, 2001.

[36] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.