Temporal Distributed Learning with Heterogeneous Data Using Gaussian Mixtures

Dean Teffer
Applied Research Laboratories
The University of Texas at Austin
Austin, U.S.A.
dteffer@arlut.utexas.edu

Amanda Hutton
Division of Statistics and Scientific Computation
The University of Texas at Austin
Austin, U.S.A.
hutton@mail.utexas.edu

Joydeep Ghosh
Department of Electrical and Computer Engg.
The University of Texas at Austin
Austin, U.S.A.
ghosh@ece.utexas.edu

Abstract – Learning a model for data in a distributed source system has often been performed by collecting all data at a central location and performing the learning process on the global data set at the central location. Although a common global feature space is normally assumed, each local source may only sample a subset of features, producing a heterogeneous data combination at the central location. Additionally, various constraints such as communication limitations and data privacy concerns require that limited information from each local source be sent to the central processor. The challenge is then to learn the most accurate global data model given this constrained information. In online systems, the data may be non-stationary, requiring explicitly dynamic modeling. We have proposed an online dynamic method to learn the probability distribution of a global data set as a Gaussian mixture model given synchronous updates of distribution parameters from local data sources of possibly non-overlapping features.

I. INTRODUCTION AND MOTIVATION

The analysis of single source data and accurately modeling the associated signal statistics and signal dynamics has provided innumerable scientific and engineering challenges over the past decades. The combination of data from multiple sources in a networked environment adds a long list of challenges [1] from state synchronization to feature mismatch to sensor registration that have confounded many data combination efforts in practice. Nonetheless, the explosion in deployed sensors in many application fields has increased the interest in combining related information [2]. Additionally, whereas batch data processing is often appropriate for static data or one-time events, much data is now available or can be updatable dynamically, increasing the interest in online learning methods. Inconsistent or inadequate communication channels as well as varying information sharing standards for each data source

require a distributed computation framework. In systems with limited local computation resources, segmenting data and associated processing to a distributed network has become increasingly popular with frameworks such as MapReduce.

This work combines elements from current research in both distributed learning and online learning to forge a temporal distributed learning method that may have broad application. The hope is that by providing a general framework, one or more of the persistent challenges in data source combination may be surmounted in follow-on research. This work assumes a processing system with local data collectors, data sources, or data processors, referred to as local sources or nodes with corresponding local models. There is also a designated central processor, which is referred to as the central node or global node with a corresponding global model. Synchronous data updates are also assumed.

This work takes the distributed learning model of [3] which fits a Gaussian mixture model (GMM) for the “global” data, which is a union of the local data. Since the local sources may have differing feature sets, the global data may be non-randomly sparse in features for which there is no data from a particular source. Each local data set’s information content can be described via a parametric model that can range from a single parameterized Gaussian component to a component for each data point. The fewer number of parameters used, the more privacy is retained, under the model in [3]. The local data can have all or a subset of the features in the global data. To accommodate non-overlapping feature spaces at the global node, the Data Augmentation method of [4] is used. Features can be missing for a variety of reasons from privacy redaction to collection differences.

For online use, data is assumed to be collected or processed at the local sources sequentially and synchronously so that the GMM parameters learned for the global model for each time step can be used as the observed data in a linear dynamical system model.

Modeling the state space dynamics of the global data space is made tractable for real online data scenarios by the following approximation. For each GMM component, a linear motion model is assumed, enabling a variant of the Kalman filter to be used to propagate model components in time. The Gaussian mixture probability hypothesis density filter [5] is a tractable form of the Kalman Filter for GMMs, that also accommodates new component formation and component death, as well as the normally challenging problem of component association on data updates for non-stationary data. This is called the multiple track problem and is called out for particular attention in [2].

A. Contributions

This work pieces together the documented algorithms and methods mentioned above and includes the appropriate glue. More than an assemblage or chaining of algorithms, the combination involves modification and alteration to all key algorithms in order to form a coherent whole. We have developed an online algorithm intended to learn the distribution parameters for non-stationary global data in a system involving heterogeneous feature sets from distributed data sources with inter-node communication limitations and/or privacy restrictions. Noise and clutter, or non-signal data that can appear to be signal, may also corrupt local data, and may be properly separated by the algorithm as these have differing temporal signatures than typical signals. At each synchronous data update, the algorithm produces an estimate of the parameters of a GMM fit to the global data.

The method is tested with a simulated datasets and out-of-sample log-likelihood is used as a quality metric, since the goal is to best fit a global parametric model to the union of a distributed set of source data.

II. BACKGROUND AND RELATED WORK

A. Distributed Learning

The Merugu-Ghosh model [3] for distributed learning is a constrained model integration method for combining data from sources with possibly non-overlapping feature spaces, without requiring all data to be shared. Due to communication restrictions, efficiency, privacy, or other constraints, only a limited amount of information can be sent from each

local source in the form of reduced feature sets and parameterized data distributions (with the case of a mixture of multivariate delta distributions, one for each datum, corresponding to just sending the data.) The set of parameters of each local distribution is used to form a global GMM parametric model by minimizing the Kullback-Leibler divergence (KL) between the (possibly projected) global and local models. Due to the relationship between KL and the log-likelihood given by equation (1), this can also be done for sufficiently large sampling sizes by minimizing the log-likelihood of the global model relative to samples drawn from the model for each local data set, which is the method adapted for this work, shown as presented in [3] in Table I.

$$\frac{1}{m} \log p_{\lambda_c}(\chi) = -KL(p_{\chi} \parallel p_{\lambda_c}^F) + H(p_{\chi}) \quad (1)$$

where λ_c is a global model, $p_{\lambda_c}(\chi)$ is the conditional distribution of the data give the model or the likelihood, $F \subset F_c$ is the global feature set, $\chi = \{x_j\}_{j=1}^m$, and $p_{\lambda_c}^F$ is the marginal distribution of p_{λ_c} on F .

TABLE I. Constrained Parametric Model Integration [3]

<p>Input: Set of models $\{\lambda_i\}_{i=1}^n$ with weights $\{\nu_i\}_{i=1}^n$ summing to 1, Parametric family G, Global sample size $\tilde{m}_c$.</p> <p>Output: $\lambda_c^a \cong \arg \min_{\lambda_c \in G} \sum_{i=1}^n \nu_i KL(p_{\lambda_i} \parallel p_{\lambda_c}^{F_i})$</p> <p>Method:</p> <ol style="list-style-type: none"> 1. Generate $\tilde{\chi}_i, [i]^n$ from the local models, $\lambda_i, [i]^n$ using MCMC sampling such that $\tilde{\chi}_i = \nu_i \tilde{m}_c, [i]^n$. 2. Let $\tilde{\chi}_c = \bigcup_{i=1}^n \tilde{\chi}_i$. Apply EM algorithm to obtain the optimal model λ_c^a such that $\lambda_c^a = \arg \max_{\lambda_c \in G} \log(p_{\lambda_c}(\chi_c))$

The derivation of this model is presented in detail in [3]. In brief, the quality of a global model λ_c is measured using log-likelihood

$$Q_{DL}(\lambda_c) = \sum_{i=1}^n \log p_{\lambda_c}(\chi_i) \quad (2)$$

Given (1) and (2), a cost function (3) is convex and can be minimized in lieu of maximizing (2).

$$C_{KL}(\lambda_c) = \sum_{i=1}^n \nu_i KL(p_{\lambda_i} \| p_{\lambda_c}^{F_i}) \quad (3)$$

The maximum entropy principle is invoked to produce a unique global model at the cost function minimum. This generates the formal Model Integration problem from [3] in (4).

$$\max_{p_{\lambda_m}} H(p_{\lambda_m}), M = \arg \min_{p_{\lambda_c} \in P(F_c)} C_{KL}(p_{\lambda_c}) \quad (4)$$

where $P(F_c)$ is the set of all probability distributions over the global feature set. In order to simplify the algorithm and to improve model interpretation, an additional constraint is applied that the global model belongs to a specified parametric family $G \subseteq P(F_c)$, such as a Gaussian mixture model. Note that this constrained model may not have a convex cost function. Sampling from the local models and in the limit of large sample sizes, the following relation holds, in which the entropy of the local model is independent of the global model so that minimizing the KL divergence with respect to the local models is equivalent to maximizing the log-likelihoods of the sampled datasets. This leads to the algorithm described in Table I.

B. Non-Overlapping Feature Spaces

Usually missing values in a model represent either the “missing” latent variables of GMM component assignment or actual missing elements in the vector of values for some data. In the data investigated here, there are both missing latent variables of GMM component assignment and missing data elements due to missing features for data from some sources. Each category of missing data impacts the estimation of the other, so a simple initial pass data imputation followed by an EM GMM parameter fit is not correct. Reference [4] describes the general method of *stochastic EM* sampling when $p(z|x)$ cannot be directly sampled, as is the case here due to z being a concatenation of dependent missing variable sets.

The appropriate model to use in this case is data augmentation described in [4]. Data augmentation prescribes an iteration of imputation – posterior, IP, in which in the I step the missing data is drawn from distributions parameterized by GMM component parameters and in the P step the GMM component parameters and the prior parameters are re-estimated given the updated missing data. See Table II for the details.

Note that since the data augmentation IP method prescribes a new way to estimate the GMM component parameters, this method replaces the EM method in Table I. Also, the KL minimization is maintained but the form of the log-likelihood optimization is modified to reflect the prior, as shown in Section 3. The prior for a multivariate Gaussian mixture is the normal-inverted Wishart for the component parameters, and a symmetric Dirichlet prior over the vector of weights as described in [7]. Note that atypically there is a set of normal-inverse Wishart hyperparameters for each mixture component. See the plate diagram of the model in Figure 1.

$$\begin{aligned} \Sigma_k &\sim W^{-1}(m_k, \Lambda_k) \\ \mu_k | \Lambda_k &\sim N(\mu_{0,k}, \tau_k^{-1} \Sigma_k) \\ w &\sim D(\alpha, \dots, \alpha) \end{aligned} \quad (5)$$

Figure 1 Plate model of GMM with parameter priors

TABLE II. Data Augmentation for GMM with Prior [4]

Initialization: Fill in missing data values with simple imputation, Seed GMM component and prior parameters with simple clustering of imputed data

IP-Iteration: Gibbs sampling past burn in period (1000 samples typical)

I-Step: Draw missing data elements from associated GMM component (ignoring index, t , indicating iteration step)

foreach $x_n, n \in 1, \dots, N$

$$k : z_{nk} = 1$$

$$x_{n,m} \sim N(\mu_{k,m|o}, \Sigma_{k,mm|o})$$

$$\mu_{k,m|o} = \mu_{k,m} + \Sigma_{k,om}^T \Sigma_{k,oo}^{-1} (x_{n,o} - \mu_{k,o})$$

$$\Sigma_{k,mm|o} = \Sigma_{k,mm} - \Sigma_{k,om}^T \Sigma_{k,oo}^{-1} \Sigma_{k,om}$$

where the missing and observed portions of the mean and covariance have been segmented.

P-Step: Estimate GMM component and prior parameters

1. Indicator variables

$$h_{nk} = \frac{w_k N(x_n | \mu_k, \Sigma_k)}{\sum_{i=1}^K w_i N(x_n | \mu_i, \Sigma_i)}, \text{ where the}$$

right-hand side parameters are from the $t-1$ step

$$z_n \sim M(h_{n1}, \dots, h_{nK})$$

2. Component parameters

(Note: data element x_n contributes to the k^{th} component indicated by $z_{nk} = 1$.)

$$N_k = \sum_n z_{nk}$$

$$\bar{x}_k = \frac{1}{N_k} \sum_n x_n \delta_{n,z_{nk}}$$

$$S_k = \frac{1}{N_k} \sum_n (x_n - \bar{x}_k)(x_n - \bar{x}_k)^T \delta_{n,z_{nk}}$$

$$m_k^t = m_k^{t-1} + N_k$$

$$\Lambda_k^t = \left(\frac{1}{\Lambda_k^{t-1}} + N_k S_k + \frac{\tau_k^{t-1} N_k}{\tau_k^{t-1} + N_k} (\bar{x}_k - \mu_{0,k}^{t-1})(\bar{x}_k - \mu_{0,k}^{t-1})^T \right)^{-1}$$

$$\mu_{0,k}^t = \frac{N_k}{\tau_k^{t-1} + N_k} \bar{x}_k + \frac{\tau_k^{t-1}}{\tau_k^{t-1} + N_k} \mu_0^{t-1}$$

$$\tau_k^t = \tau_k^{t-1} + N_k$$

$$\Sigma_k \sim W^{-1}(m_k, \Lambda_k)$$

$$\mu_k | \Lambda_k \sim N(\mu_{0,k}, \tau_k^{-1} \Sigma_k)$$

3. Mixing coefficients

$$w^t \sim D(\alpha + N_1, \dots, \alpha + N_K)$$

Output: GMM component, prior parameters, missing data elements

$$\{\mu, \Sigma\}, \{m, \Lambda, \mu_0, \tau\}, \{x_{n,m}\}$$

C. Temporal Learning

1) Markov State Space Dynamic Model

Several approaches to temporal learning are phrased in terms of “target tracking” in which the most prominent or signal-like distributed components at each time step are either targets or clutter. Following components or targets over time is thus target tracking. Most tracking methods refer to a target dynamic model in which X_t is the set of states at time t and Z_t is the set of observations at time t with the following relationship that upholds the Markov property and the additional property that target motion is linear:

$$\begin{aligned} p(x_t | x_{t-1}) &= N(F_t x_{t-1}, Q_t) \\ p(z_t | x_t) &= N(H_t x_t, R_t) \end{aligned} \quad (6)$$

where F and H are the usual state and observation transition matrices, and Q and R are the corresponding noise matrices.

2) Kalman Filter

The above linear dynamic system model and Markov assumption enables a set of update equations at each time step, which form the well-known Kalman filter. The equations of the Kalman filter are used here as

part of a more complex filter and documented in Table III.

Although very wide and successfully used, the Kalman filter suffers from multiple shortcomings, from the invalidity of the linear dynamic model for many real systems to difficulties with tracking multiple targets at the same time. This latter issue has proven challenging for temporal models in general [2].

3) Probability Hypothesis Density Filter

One approach to the multiple target tracking problem is the Gaussian Mixture Probability Hypothesis Density (PHD) filter. Reference [5] describes an implementation of this approach for data modeled as a Gaussian Mixture Model, GMM, at each time update. Briefly, the PHD filter is related to the zero-gain Kalman filter in that it is mean-only tracker. Thus, at each time step, only the mean of each GMM component is updated, and the component covariance is kept constant and equal to the initial value. The PHD filter forms new tracks for each component at each data update and uses the constant covariance simplification to make data assignments to the proper track in its list, new or prior. A separate pruning step is required to merge tracks likely to represent the same component, which is inevitable given the forced track creation at the beginning of the processing loop. Since the output of our single-update global data learning algorithm is a GMM, the PHD filter may be a reasonable first step at introducing a temporal model to distributed learning. Since only a single line is added to each of the two PHD algorithms required from [5], these are referenced here without listing as Table III. The modification is discussed in Section 3.

TABLE III. Gaussian Mixture Probability Hypothesis Density Filter [5]

Table I: Pseudocode for the Gaussian Mixture PHD Filter
Table II: Pruning for the Gaussian Mixture PHD Filter

III. TEMPORAL DISTRIBUTED LEARNING

A. Recursive Bayesian Estimation

Equation (7) describes the recursive Bayesian estimation problem when updates are provided at each t .

$$p(x_t | z_t) = kp(z_t | x_t) \int_{x_{t-1}} p(x_t | x_{t-1}) p(x_{t-1} | z_{t-1}) \quad (7)$$

TABLE IV. Distributed Learning with Data Augmentation

(Note: In this implementation, all models from parametric family G)

Input: Set of models $\{\lambda_i\}_{i=1}^n$ with weights $\{\nu_i\}_{i=1}^n$ summing to 1, Parametric family G , Global sample size $\tilde{m}_c$.

Output: $\lambda_c^a \equiv \arg \min_{\lambda_c \in G} \sum_{i=1}^n \nu_i KL(p_{\lambda_i} \| p_{\lambda_c}^{F_i})$

Method:

1. Generate $\tilde{\chi}_i, [\tilde{i}]^n$ from the local models, $\lambda_i, [i]^n$ using Gibbs sampling such that $\left| \tilde{\chi}_i = \nu_i \tilde{m}_c, [\tilde{i}]^n, \left| \tilde{\chi} \right| > 1000 \right|$
2. Let $\tilde{\chi}_c = \bigcup_{i=1}^n \tilde{\chi}_i$. Apply data augmentation IP algorithm to obtain the optimal model λ_c^a such that $\lambda_c^a = \arg \max_{\lambda_c \in G} \log(p_{\lambda_c}(\chi_c)) + \log(\lambda_c)$

This enables the calculation of the state of system at time t given observations of a system at that time with only information from the immediately prior time step available, fulfilling the need for an on-line method. In practice, the right hand side of the equation is extremely challenging to calculate, so a wide variety of approximation methods have been developed, perhaps the most historically popular of which, the Kalman filter, was discussed in the prior section. Regardless of the approximation method used, in fielded applications with even single sensors, complications arise due to myriad causes including the aforementioned multiple target tracking, very large numbers of targets to track, clutter (signal-like non-targets), noise, false and spurious (single or few update) targets, dropouts (the inverse of spurious targets), and other algorithm –confounding data issues.

B. Combining Heterogeneous Data

A close examination of Table I reveals that in step 2, if the data is heterogeneous, there will be large segments of missing values. For example consider the exemplar scenario from [3] in which 3 heterogeneous data sources with feature sets (A,B), (B,C), and (C,A) are combined into a global feature space (A,B,C). In the Distributed Learning algorithm when samples are combined, set 1 will lack feature C,

and so on. Thus if the data to be processed via EM is vector data, the 3-D vector data will have one section with C-missing values, one with A-missing values, and one with B-missing values. As previously discussed, performing EM to fit the missing values of GMM parameters simultaneously with fitting these missing data values are correlated problems. Thus the data augmentation IP method described in Table I must be used. Since the output of this algorithm is a GMM model fit to the input data, a minor change to Table I is motivated. This is described in Table IV. For simplicity of analysis and evaluation, the local models are also assumed to be taken from the same parametric family as the global model, namely the Gaussian mixture model family. Note also that a prior term has been added to the optimization as required by the data augmentation algorithm.

C. Tracking Mixture Components

Since as noted previously the PHD filter is a constant covariance filter in which only mixture component means are updated, a method is required to provide for the update of component covariance. We have chosen simply to use the learned global covariance, or the observation model component variance. This coupled with the estimated component state mean and estimated component state weight form the parameter tuple for each Gaussian mixture component produced by the method for each data update. As noted in Table V, the state and observation components are related via a correlation variable that is maintained in the updated form of the PHD filter and is a matrix form of the usual binary-valued GMM component assignment variable.

TABLE V. Modified GMM for PHD

<p>Input: Z_n, set of “observation” samples for latest data update from global model output by Table IV., Distributed Learning with Data Augmentation such that missing feature data has been filled in. These are the means of the global GMM components in λ_c^a, the global model for latest data update output by Table IV.</p>
<p>Output: $\{w^{(j)}, m^{(j)}, \Sigma^{(j)}\}_{j=1}^J$ weights, means, in state space and covariances in observation space for latest data update</p>
<p>$\{\delta^{(j)(m)}\}_{j=1, m=1}^{J, M}$ most likely correlate between observation components and state components updated state space model parameters</p>
<p>$\{H^{(j)}, Q^{(j)}, R^{(j)}\}_{j=1}^J$ for each state component</p>

steps 1 - 4: PHD GMM algorithm as described in Table III.

Insert prior to the second to final “end” and at the same level as the $\{w^{(j)}, m^{(j)}, P^{(j)}\}$ assignments for each z :

(Note: m is the component index of element z .)

$$\delta^{(j)(m)} = 1$$

Pruning: As described in Table III

Set $J_{\max} = n$ (i.e. number of state components to equal number of observation components)

Modify m in each $\delta^{(j)(m)}$ as necessary as components are grouped.

D. Implementation Notes

All algorithms were implemented in CPython with numpy and matplotlib.pyplot for figure generation.

IV. EXPERIMENTS

A. Datasets

Since this is very early stage algorithmic development a data simulator was built in order to achieve the goal of being able to examine performance on data with specific qualities. The simulator also facilitates algorithm development since multiple source, multiple update data is difficult to obtain in the research setting. Once the method has matured, standard multisource heterogeneous non-stationary data can be processed.

1) Simulated Dataset 1 – Overlapping Components, No Missing Data, Three Sources, Three Initial Clusters, Non-Stationary

The initial dataset examined covers a global 2-D space with three 2-D data sources. All sources have essentially no observation spherical noise, 0.1. Separate simulation studies have demonstrated the ability of the algorithm to reproduce the results of [3] for single data updates by introducing widely varying observation noise for each source. In this case the dataset runs from $t = 0$ to $t = 9$ or ten time steps, and there are three components to the data in all steps. At $t = 0$, the means are at (0,0), (4,4), and (8,4) and at $t = 9$, they are at (0,0), (1.3,1.3), and (8,4). That is, the second component travels with a constant velocity diagonally toward the origin. Due to component covariance, at $t = 9$, components one and two are visibly indistinguishable. Figure 1 shows the simulated dataset and distributed learning fit at $t=0$.

Figure 2 Simulated dataset (union of local data) at $t=0$ with corresponding global parameter fit using the Distributed model

Figure 3 Out-of-Sample Log-Likelihood for Compared Models (Central – Blue, Distributed – Green, Distributed Temporal – Red)

B. Quantitative Metrics

Since the veracity of alternate models is the principal question in this work, out-of-sample log likelihood, OOSLL, was used to evaluate models. Three models were examined: (i) the Central model in which all data was able to be collected at a central processor at each update and processed there, in the standard way with a EM GMM parameter fit; (ii) the model presented here, called the Distributed Temporal model; and (iii), a model in which distributed processing was performed at each step without a temporal model, called the Distributed model. Figure 3 shows the performances obtained on the simulated dataset. Since in this simple example the model performances were similar, we summed the difference at each data update in the OOSLL to a reference number which we chose to be the worst or lowest score. Thus the larger the sum or area, the better the overall fit over the course of all data updates. The data is shown in Table VI. This matches the visual impression from Figure 3 which is that the Distributed model alone is marginally better than the Central model, and that despite a couple of poor results, the overall performance of the Distributed Temporal model is superior, though by this metric only by less than 2%.

TABLE VI. Temporal Metric Comparison for Models

Model	Out-of-sample Likelihood difference	Log time sum
Central	330,845	
Distributed	366,609	
Distributed Temporal	380,016	

C. Temporal Distributed Learning Results

Examining the distribution parameter estimates at each data update rather than the quality metric, it can be seen that as the second data cluster moves progressively “into” the covariance ellipse of the first cluster, the Central and Distributed models lose differentiation earlier than the Distributed Temporal model. This is one easy to conceptualize benefit of component “tracking.” However, the PHD filter appears to require data dependent tuning to determine track merge and prune. The metric results for the Distributed Temporal model would have been much better in this example if for the last data update the easily noted third component had not been assigned a vanishingly low weight by the PHD filter. It is possible that additional improvements to this filter or an alternate approach such as sequential Monte Carlo may provide more accurate parameter estimates of the global data distribution.

V. CONCLUSIONS

A temporal model has been proposed to estimate non-stationary GMM component parameters in a global feature space derived from distributed heterogeneous data sources. Each source passes model parameters rather than entire data sets, thus enabling operation in communication and privacy limited distributed environments. Our next steps are to incorporate the data augmentation IP model for heterogeneous data and to study such datasets using a PHD filter with a reduced component set produced via improved pruning or using a sequential Monte Carlo filter.

ACKNOWLEDGEMENT

This research was supported in part by a grant from TATP.

REFERENCES

- [1] Kumar, P.R. "Challenges in Cyberphysical Systems," University of Texas at Austin, Department of Electrical and Computer Engineering, Invited Talk, October, 2010.
- [2] Juan Liu, Maurice Chu, and James Reich, "Multitarget Tracking in Distributed Sensor Networks," *IEEE SIGNAL PROCESSING MAGAZINE*, pp. 36-46, May 2007.
- [3] Merugu and Ghosh, "A distributed learning framework for heterogeneous data sources," *Proceedings of Knowledge Discovery and Data Mining*, pp. 208-217, 2005.
- [4] Jufen Zhang and R. Everson, "Bayesian Estimation and Classification with Incomplete Data," in *International Conference on Machine Learning and Applications, 2004. Proceedings. 2004*, 2004, pp. 296-303.
- [5] Ba-Ngu Vo and Wing-Kin Ma, "The Gaussian Mixture Probability Hypothesis Density Filter," *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, vol. 54, no. 11, pp. 4091-4104, November 2006.
- [6] Christopher Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
- [7] Chris Fraley and Adrian Raftery, "Bayesian Regularization for Normal Mixture Estimation and Model-Based Clustering," Department of Statistics, University of Washington, Seattle, WA, Technical Report 486, 2009.