A Look At Gaussian Mixture Reduction Algorithms

David F. Crouse, Peter Willett, and Krishna Pattipati
Department of Electrical and Computer Engineering
University of Connecticut
371 Fairfield Way, U-2157
Storrs, CT 06269, USA
Email: {crouse, willett, krishna}@engr.uconn.edu

Lennart Svensson
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Göteborg, Sweden
Email: lennart.svensson@chalmers.se

***Abstract*—We review the literature and look at two of the best algorithms for Gaussian mixture reduction, the GMRC (Gaussian Mixture Reduction via Clustering) and the COWA (Constraint Optimized Weight Adaptation) which has never been compared to the GMRC. We note situations that could yield invalid results (i.e., reduced mixtures having negative weight components) and offer corrections to this problem. We also generalize the GMRC to work with vector distributions. We then derive a brute-force approach to mixture reduction that can be used as a basis for comparison against other algorithms on small problems. The algorithms described in this paper can be used in a number of different domains. We compare the performance of the aforementioned algorithms along with a simpler algorithm by Runnalls' for reducing random mixtures, as well as when used in a Gaussian mixture reduction-based tracking algorithm.**

Keywords: Gaussian mixture reduction, nonlinear optimization, clustering, ISE, tracking

I. Introduction

Gaussian mixture reduction algorithms are useful in many areas, including in error correction codes [1], supervised learning of multimedia data [2], distributed data fusion [3], and pattern recognition [4]. In many target tracking problems, data association uncertainty manifests itself as multiple hypotheses, each corresponding to a component in a multivariate Gaussian mixture posterior distribution. It is naturally of interest to reduce their number with minimal loss of fidelity. The simplest method of managing the complexity is to eliminate Gaussian components having low probabilities, as used, for example, in the track or hypothesis-oriented Multi-Hypothesis Tracker (MHT) [5]. Alternatively, instead of pruning components of the Gaussian mixture, one could merge components according to a similarity measure. Hypothesis merging can be considered more attractive than (MHT-style) pruning, since the information lost in removing mixture elements is, in some sense, preserved in the uncertainty (covariances) of those retained. Salmond is among the first to consider a tracker based on merging hypotheses according to an *ad hoc* similarity measure [6].[1] Pao considered the generalization of Salmond's tracker to the multisensor-multitarget case [8]. Such trackers, which are based upon hypothesis merging, may be considered to be variants of the hypothesis-oriented MHT or as multimodal generalizations of the Joint Probabilistic Data Association Filter (JPDAF), as described, for example, in [9].

More advanced techniques for Gaussian mixture reduction optimize the parameters of the reduced mixture according to a global optimality criterion.[2] One of the first uses of optimization-based mixture reduction was by Scott and Szewczyk [11], [12], who introduced the so-called L_2E measure of similarity as well as a correlation measure of similarity. The L_2E measure is more commonly known as the Integral Squared Error (ISE).[3] The correlation measure was also derived by Petrucci in the context of target tracking [10], highlighting its relationship to the L_2E and has more recently been considered by Harmse [13], who considered accelerating clustering with scalar states. The ISE measure was presented in the context of tracking by Williams and Maybeck [14], [15]. The ISE has been around since at least the 1950's [16], but not in the context of Gaussian mixture reduction for tracking. A Gaussian mixture reduction algorithm utilizing the ISE, similar to that of Williams and Maybeck, was derived by Kurkoski and Dauwels [1] in the context of error correcting codes.

Many mixture reduction algorithms, such as by Williams and Maybeck [14], Petrucci [10] and Scott and Szewczyk [12], are either greedy in nature, or are initialized by using a greedy algorithm that often bears similarities to West's algorithm for mixture reduction [17]. The greedy initialization algorithm is important, because the criteria of optimality are all multimodal. As an alternative to Williams and Maybeck's greedy method, sequentially optimizing over the ISE, Huber and Hanebeck [18] introduced an optimization utilizing a homotopy in order to reduce the chances of converging to a local minimum [18]. Rather than starting with the full mixture and removing or merging components, this method creates a mixture from one component and then sequentially adds new components. The reasoning behind the method is similar to that used in algorithms utilizing simulated annealing and can be thought of as an extension of the work by Schrempf, Feiermann and Hanebeck using progressive Bayes estimation [19]. Schieferdecker and Huber later demonstrated [20] that the method could be beaten in terms of the ISE by

This work was partially supported by the Office of Naval Research under contracts N00014-09-10613 and N00014-10-10412.

[1]This journal article was preceded by a conference paper in 1990. [7]

[2]A detailed comparison of the optimality criteria is given in [10].

[3]The ISE, as named in [11], among other papers, is sometimes called the Integral Squared Difference or the Integral Squared Distance [10].

978-0-9824438-2-8/11/$26.00 ©2011 IEEE

the GMRC algorithm. The GMRC algorithm uses a greedy approach developed by Runnalls [21], which minimizes an upper bound on the Kullback-Leiber (KL) divergence between the full mixture and the reduced mixture, followed by a k-means clustering using the KL divergence between the cluster centers and the merged states as a distance measure and finally followed by an iterative optimization. The use of the k-means algorithm for Gaussian mixture reduction using the KL-divergence as the distance measure had previously been considered by Nikseresht and Gelgon [2] in the context of pattern recognition. Chen, Chang and Smith [3] presented an algorithm, the COWA, utilizing a modified form of Williams algorithm, followed by a refinement step on the mixture weights, different from that used in the GMRC by Schieferdecker and Huber. The COWA was later refined by Chang and Sun [22] to improve the performance and to better estimate the best number of components to use in the reduced mixture.

Algorithms based upon the expectation maximization algorithm and variational Bayes estimation have been considered, respectively by Petrucci [10], and by Bruneau, Gelgon and Picarougne [4]. However, Petrucci showed that the EM solution is slow and unreliable. On the other hand, the variational Bayes method can not be used "out-of-the-box" for general problems, since a number of prior PDFs must have their parameters tuned (in a trial-and-error manner) for each new situation. Gaussian mixture reduction could also be performed by sampling the full mixture and fitting the parameters of the reduced mixture to the samples using one of many pattern recognition algorithms [23].

In Section II, we define the mixture reduction problem as well as optimality criteria, including the ISE measure. In Section III, we present the GMRC and the COWA algorithms, generalizing the GMRC to handle multidimensional mixtures and correcting a problem in both algorithms where the mixture weights could be invalid (negative). In Section IV, we derive a better performing, but also computationally more expensive, brute-force algorithm that can be used as a basis for comparison with more practical mixture reduction algorithms. Section V compares the performance of the modified GMRC, COWA and the brute force method both on randomly generated mixtures, as well as in a practical tracking scenario. We conclude in Section VI.

II. Optimization Criteria for the Gaussian Mixture Reduction Problem

Let our initial mixture PDF consist of N D-dimensional multivariate Gaussian components, the i^{th} one having weight w_i, such that all $w_i > 0$ and $\sum_{i=0}^{N} w_i = 1$, and means and covariance μ_i and P_i, which we shall denote as $\mathbf{\Omega}_N$. The reduced mixture shall consist of L components, the i^{th} one having weight, mean, and covariance $\tilde{w}_i$, $\tilde{\mu}_i$, and $\tilde{P}_i$, which shall be denoted as $\mathbf{\Omega}_L$. The PDFs are thus:

$$f(\mathbf{x}|\mathbf{\Omega}_N) = \mathbf{w}^T \mathbf{N}(\mathbf{x}) \qquad f(\mathbf{x}|\mathbf{\Omega}_L) = \tilde{\mathbf{w}}^T \tilde{\mathbf{N}}(\mathbf{x}) \tag{1}$$

where

$$\mathbf{w} = [w_1, w_2, \ldots, w_N]^T \qquad \tilde{\mathbf{w}} = [\tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_L]^T \tag{2}$$

$$\mathbf{N}(\mathbf{x}) = [\mathcal{N}\{\mathbf{x}; \mu_1, P_1\}, \ldots, \mathcal{N}\{\mathbf{x}; \mu_N, P_N\}]^T \tag{3}$$

$$\tilde{\mathbf{N}}(\mathbf{x}) = \left[\mathcal{N}\left\{\mathbf{x}; \tilde{\mu}_1, \tilde{P}_1\right\}, \ldots, \mathcal{N}\left\{\mathbf{x}; \tilde{\mu}_L, \tilde{P}_L\right\}\right]^T \tag{4}$$

$$\mathcal{N}\{\mathbf{x}; \mu_i, P_i\} \triangleq |2\pi P_i|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}-\mu_i)^T P_i^{-1}(\mathbf{x}-\mu_i)} \tag{5}$$

We would like to determine the parameters of the reduced mixture, $f(\mathbf{x}|\mathbf{\Omega}_L)$, such that the "similarity" between the original and the reduced mixture is maximized. The Integral Squared Error (ISE) cost function, initially presented in the context of tracking in [15], is the most widely used metric for Gaussian mixture reduction, since (unlike some that might be more attractive in terms of their statistical implications) it can be expressed in a simple and closed-form. The ISE is given by

$$J_S(\mathbf{\Omega}_L) \triangleq \int_x \left(f(\mathbf{x}|\mathbf{\Omega}_N) - f(\mathbf{x}|\mathbf{\Omega}_L)\right)^2 d\mathbf{x} \tag{6}$$

$$= \tilde{\mathbf{w}}^T H_1 \tilde{\mathbf{w}} - 2\mathbf{w}^T H_2 \tilde{\mathbf{w}} + \mathbf{w}^T H_3 \mathbf{w} \tag{7}$$

$$= J_{LL} - 2J_{NL} + J_{NN} \tag{8}$$

where

$$H_1 = \int_{\mathbf{x}} \tilde{\mathbf{N}}(\mathbf{x}) \tilde{\mathbf{N}}(\mathbf{x})^T d\mathbf{x} \tag{9}$$

$$H_2 = \int_{\mathbf{x}} \mathbf{N}(\mathbf{x}) \tilde{\mathbf{N}}(\mathbf{x})^T d\mathbf{x} \tag{10}$$

$$H_3 = \int_{\mathbf{x}} \mathbf{N}(\mathbf{x}) \mathbf{N}(\mathbf{x})^T d\mathbf{x} \tag{11}$$

The elements of H_1, H_2 and H_3 may be evaluated directly by noting that integral of the product of two Gaussian PDFs is a constant times a Gaussian PDF, given as:

$$\int_{\mathbf{x}} \mathcal{N}\{\mathbf{x}; \mu_i, P_i\} \mathcal{N}\{\mathbf{x}; \mu_j, P_j\} \, d\mathbf{x} = \mathcal{N}\{\mu_i; \mu_j, P_i + P_j\} \tag{12}$$

A more detailed derivation of the integral of the product of two Gaussian PDFs is given in [15]. The entries in the i^{th} row and j^{th} column of H_1 and H_2 are respectively $\mathcal{N}\left\{\tilde{\mu}_i; \tilde{\mu}_j, \tilde{P}_i + \tilde{P}_j\right\}$ and $\mathcal{N}\left\{\mu_i; \tilde{\mu}_j, P_i + \tilde{P}_j\right\}$. The value J_{NN} does not depend upon the parameters over which the optimization is being performed. The terms of (8) may thus be expressed compactly as

$$J_{LL} = \sum_{l_1=1}^{L} \sum_{l_2=1}^{L} \tilde{w}_{l_1} \tilde{w}_{l_2} \mathcal{N}\left\{\tilde{\mu}_{l_1}; \tilde{\mu}_{l_2}, \tilde{P}_{l_1} + \tilde{P}_{l_2}\right\} \tag{13}$$

$$J_{NL} = \sum_{n=1}^{N} \sum_{l=1}^{L} w_n \tilde{w}_l \mathcal{N}\left\{\mu_n; \tilde{\mu}_l, P_n + \tilde{P}_l\right\} \tag{14}$$

$$J_{NN} = \sum_{n_1=1}^{N} \sum_{n_2=1}^{N} w_{n_1} w_{n_2} \mathcal{N}\{\mu_{n_1}; \mu_{n_2}, P_{n_1} + P_{n_2}\} \tag{15}$$

Other measures of similarity for Gaussian mixture reduction have been extensively studied by Petrucci [10]. For example, the Normalized ISE (NISE) can only vary between zero and one,

$$J_{SN}(\mathbf{\Omega}_L) = \frac{\int_{\mathbf{x}} \left(f(\mathbf{x}|\mathbf{\Omega}_N) - f(\mathbf{x}|\mathbf{\Omega}_L)\right)^2 d\mathbf{x}}{\int_{\mathbf{x}} f(\mathbf{x}|\mathbf{\Omega}_N)^2 \, d\mathbf{x} + \int_{\mathbf{x}} f(\mathbf{x}|\mathbf{\Omega}_L)^2 \, d\mathbf{x}} \tag{16}$$

$$= \frac{J_{NN} - 2J_{NL} + J_{LL}}{J_{NN} + J_{LL}} \tag{17}$$

whereas Petrucci [10] and Scott used a related correlation measure [11], which should be maximized.[4]

Since no closed-form solution for the optimal parameters of the reduced mixture has been found, the GMRC and COWA algorithms utilize a number of methods to approximate the optimally reduced mixture before iteratively minimizing ISE as a function of the reduced mixture parameters. Since the ISE cost function can have many local minima, having a good initial estimate is key. We shall discuss the GMRC and COWA in the following section.

III. The GMRC, the COWA, and Improvements

The GMRC [20] algorithm was conceived with scalar variables in mind ($D = 1$). However, we shall consider everything in a vector context. The GMRC works as follows:

Algorithm 1: The GMRC

1) (Greedy Initialization) Run Runnalls' algorithm (Algorithm 3 in Subsection III-A) to get an initial estimate of $f(\mathbf{x}|\mathbf{\Omega}_L)$.
2) (Clustering) Run the k-means algorithm with $k = L$ using the Kullback-Leiber divergence between Gaussians in the mixture as the distance measure to refine the estimates.
3) (Refinement) Perform iterative optimization over the ISE measure to refine the estimates.

The refined COWA [22] algorithm is intended to be used both for mixture reduction as well as order determination. The algorithm is normally run with an error bound ϵ, a minimum number of components L, and a distance threshold γ. The algorithm works as follows:

Algorithm 2: The COWA

1) (Greedy Reduction) Run the enhanced West algorithm (Algorithm 4 in Subsection III-A) to reduce the mixture by one component. If γ is such that the mixture can not be reduced, then the algorithm is finished.
2) (Refinement) Given the estimates of the means and covariances of the reduced mixture, calculate the globally optimal mixture weights as described in Section III-C.
3) (Further Reduction) If the reduced mixture has L components, or if the ISE of the reduced mixture is below ϵ, then stop; otherwise, go back to 1.

The original version of the COWA from [3] implicitly used $\gamma = \infty$ in the enhanced West algorithm.

Both algorithms use various techniques to generate an approximation to the optimal reduced mixture prior, which is then used as an initial estimate in a refinement step that optimizes over the ISE. In Subsection III-A, we discuss greedy algorithms for mixture reduction, specifically describing Runnalls' algorithm, used in the GMRC; and the enhanced West algorithm, used in the COWA. In Subsection III-B, we then explain how the k-means algorithm is used in the GMRC. Finally, in Subsection III-C, we discuss the optimization performed in the final steps of both algorithms, and how it can be improved. Namely, we note that we do not have to perform optimization over all D^2 elements of each covariance matrix, and we add constraints to the weight optimization to assure that all weights (i.e., $\tilde{w}$) are positive and sum to one. These improvements have not been considered in past work. Additionally, we will note that unlike in previous work, our presentation of the GMRC is generalized for vector distributions.

A. Greedy Initialization Algorithms

Runnalls' algorithm is one of a number of greedy initialization algorithms for mixture reduction. Most greedy algorithms for Gaussian mixture reduction try to decide which components of a Gaussian mixture should be merged or pruned in order to reduce the mixture to the desired number of components. The exception to this is the algorithm by Huber and Hanebeck [18], which continually adds new components to a mixture beginning with a single component. However, simpler approaches have proven to have better performance [20].

Pruning is the simplest approach to mixture reduction. Given a Gaussian mixture consisting of N Gaussians, one can discard the $N - L$ components having the lowest cost (according to some measure), and then renormalize the weights of the remaining components (i.e., the $\tilde{w}$'s must sum to one). However, this pruning has proven inferior to more sophisticated greedy methods [14].

Instead of pruning, one can perform mixture reduction utilizing merging, whereby the merging of the components comes from taking an expected value across the set of components that are to be merged, similar to how the JPDAF functions. This preserves the moments of the overall mixture. For example, if one wished to merge components 1 to C, then the equations for the merged result are

$$w_{\text{merged}} = \sum_{i=1}^{C} w_i \tag{18}$$

$$\mu_{\text{merged}} = \frac{1}{w_{\text{merged}}} \sum_{i=1}^{C} w_i \mu_i \tag{19}$$

$$P_{\text{merged}} = \sum_{i=1}^{C} \frac{w_i}{w_{\text{merged}}} \left(P_i + (\mu_i - \mu_{\text{merged}})(\mu_i - \mu_{\text{merged}})^T \right) \tag{20}$$

Greedy optimization methods differ in how they choose which components are to be merged. Williams and Maybeck's algorithm [14] is one of the best, but also has been observed to be more computationally complex than other approaches [20]. Runnalls' greedy algorithm [21] minimizes an upper bound on the increase in the Kullback Leibler divergence between the original and the reduced mixtures. It is used in the GMRC algorithm and is given below:

[4] Petrucci [10] observed an (almost negligible) advantage for optimization over the correlation measure when used in a tracking scenario. In this paper, as in the GMRC, we will only consider optimization over the ISE.

Algorithm 3: Runnalls' Algorithm

1) Set the current mixture to the full mixture.
2) The cost of merging components i and j in the current mixture is the upper bound on the increase in the KL divergence [21],

$$c_{i,j} = \frac{1}{2}\left((w_i + w_j)\log\left[|P_{i,j}|\right] - w_i \log\left[|P_i|\right]\right) - \frac{1}{2} w_j \log\left[|P_j|\right] \quad (21)$$

where $P_{i,j}$ corresponds to (20) for merging only components i and j. Merge the components of the current mixture having the lowest $c_{i,j}$ using (18)–(20) and set the current mixture to the result.
3) If the current mixture has the desired number of components, quit. Otherwise, go back to 2.

The algorithms by Salmond [6] and Pao [8] are the same as Runnalls', but they use a different cost measure and have been shown to perform worse than Runnalls' method [21]. West's algorithm [17] is similar, except rather than considering all possible mergers at each step, it only considers mergers involving the component with the smallest weight. The "enhanced" West algorithm, used in the COWA [3], is a variant on that. The version presented in [22] is the same if the distance threshold $\gamma = \infty$. The algorithm is used in the COWA to reduce the mixture by one component; it is given as follows:

Algorithm 4: The Enhanced West Algorithm

1) Calculate a set of modified weights for the components of the current mixture:

$$\tilde{w}_i^{\text{mod}} = \frac{\tilde{w}_i}{\text{trace}\left[P_i\right]} \quad (22)$$

2) Let i denote the component with the smallest modified weight. The cost of merging component j with component i is given by the ISE distance between them:

$$c_{i,j} = -2\mathcal{N}\left\{\mu_j; \mu_i, P_j + P_i\right\} + \sum_{k\in\{i,j\}} \mathcal{N}\left\{\mu_k; \mu_k, 2P_k\right\} \quad (23)$$

Choose i and j such that i corresponds to the component having the smallest modified weight such that $c_{i,j} < \gamma \,\forall j \neq i$[5] and j is chosen such that $c_{i,j}$ is minimized for this i. If no such pair can be found, then the reduction is complete. Otherwise, merge the components corresponding to the smallest $c_{i,j}$ using (18)–(20), and set the current mixture to the result.

B. The k-Means Algorithm for Mixture Reduction

The k-means algorithm, used in the GMRC to refine an initial estimate, was considered both by Schieferdecker and Huber [20] in the GMRC, as well as separately by Nikseresht and Gelgon [2]. The algorithm is as follows:

[5]The threshold γ is meant to prevent components that are too "distinct" from being merged. If $\gamma = \infty$, then the algorithm merges the component having the smallest weight with whichever component is the closest.

Algorithm 5: The k-Means Algorithm

1) Calculate an initial estimate of the reduced mixture, $f(\mathbf{x}|\mathbf{\Omega}_L)$, using, for example, Algorithm 3 or 4.
2) For each of the N components of the original mixture, determine which component of the reduced mixture is closest, whereby the distance between the i^{th} component of the original mixture and the j^{th} component of the reduced mixture is

$$d_{KL}(i,j) = \int_{\mathbf{x}} \mathcal{N}\left\{\mathbf{x}; \tilde{\mu}_j, \tilde{P}_j\right\} \log\left[\frac{\mathcal{N}\left\{\mathbf{x}; \tilde{\mu}_j, \tilde{P}_j\right\}}{\mathcal{N}\{\mathbf{x}; \mu_i, P_i\}}\right] d\mathbf{x} \quad (24)$$

$$= \text{trace}\left[\tilde{P}_j^{-1}\left(P_i - \tilde{P}_j + (\mu_i - \tilde{\mu}_j)(\mu_i - \tilde{\mu}_j)^T\right)\right] + \log\left[\frac{\det \tilde{P}_j}{\det P_i}\right] \quad (25)$$

3) The components in the original mixture closest to the same component in the reduced mixture form a cluster. Using (18)–(20), merge all of the components in each cluster to form a new reduced mixture.
4) If the new reduced mixture is the same as the old reduced mixture or the maximum number of iterations has elapsed, terminate. Otherwise, set the reduced mixture to the new reduced mixture and go to 2.

It should be noted that methods exist for accelerating the k-means algorithm using kd-trees [24].[6] However, the optimization over the ISE measure, described in the following subsection, is generally the slowest step in the GMRC algorithm.

C. Optimizing over the ISE Measure

We would like to maximize the similarity of the reduced PDF to the original PDF, meaning that we would like to minimize the ISE. Given an initial estimate of the weights, means and covariances of the reduced mixture (the $\tilde{w}$, $\tilde{\mu}$ and $\tilde{P}$ terms), past work has then performed explicit optimization over the ISE. Williams and Maybeck [14] performed this optimization via a Quasi-Newton method [25], whereby the gradients of the terms being optimized are needed, but second order information (the Hessian) is not needed. We shall use this method for the implementation of the third step (in a generalized vector context) in the GMRC. The gradient of the cost function in (8) is

$$\nabla J_S(\mathbf{\Omega}_L) = -2\nabla J_{NL} + \nabla J_{LL} \quad (26)$$

During the optimization, the covariance matrices must be symmetric and positive definite to remain valid. To enforce

[6]Often, the cost of setting up the data structure will exceed the performance gain elsewhere if care is not taken when allocating and deallocating the structure. In other words, one should call a memory allocation routine, such as `malloc` if programming in C, once to allocate the whole structure and not repeatedly for every node. This can be problematic when using an interpreted language, such as MATLAB, that lacks precise control over data allocation.

this constraint, Williams and Maybeck [14] performed optimization over the square root of the covariances rather than the covariances themselves. That is, optimization was performed over the set of L_i such that

$$\tilde{P}_i = L_i L_i^T \tag{27}$$

where the initial L_i may be obtained by taking the lower triangular matrix from the Cholesky decomposition (see [26]) of the initial estimate of $\tilde{P}_i$. The necessary gradients for performing optimization over the $\tilde{\mu}$ and L terms are [15]

$$\nabla_{\tilde{\mu}_j} J_{LL} = -2\tilde{w}_j \sum_{i=1}^{L} \tilde{w}_i \left(\tilde{P}_i + \tilde{P}_j\right)^{-1} \cdot (\tilde{\mu}_j - \tilde{\mu}_i)\mathcal{N}\left\{\tilde{\mu}_i; \tilde{\mu}_j, \tilde{P}_i + \tilde{P}_j\right\} \tag{28}$$

$$\nabla_{\tilde{\mu}_j} J_{NL} = -\tilde{w}_j \sum_{i=1}^{N} w_i \left(P_i + \tilde{P}_j\right)^{-1} \cdot (\tilde{\mu}_j - \mu_i)\mathcal{N}\left\{\mu_i; \tilde{\mu}_j, P_i + \tilde{P}_j\right\} \tag{29}$$

$$\nabla_{L_j} J_{LL} = 2\sum_{i=1}^{L} \tilde{w}_i \tilde{w}_j \mathcal{N}\left\{\tilde{\mu}_i; \tilde{\mu}_j, \tilde{P}_i + \tilde{P}_j\right\} \left(\tilde{P}_i + \tilde{P}_j\right)^{-1} \cdot \left((\tilde{\mu}_i - \tilde{\mu}_j)(\tilde{\mu}_i - \tilde{\mu}_j)^T - \left(\tilde{P}_i + \tilde{P}_j\right)\right) \left(\tilde{P}_i + \tilde{P}_j\right)^{-1} L_j \tag{30}$$

$$\nabla_{L_j} J_{NL} = \sum_{i=1}^{N} w_i \tilde{w}_j \mathcal{N}\left\{\mu_i; \tilde{\mu}_j, P_i + \tilde{P}_j\right\} \left(P_i + \tilde{P}_j\right)^{-1} \cdot \left((\mu_i - \tilde{\mu}_j)(\mu_i - \tilde{\mu}_j)^T - \left(P_i + \tilde{P}_j\right)\right) \left(P_i + \tilde{P}_j\right)^{-1} L_j \tag{31}$$

The optimization does not need to be performed over every element of the L_i matrices, rather, only the lower triangular elements of each L_i need be optimized, with the upper triangular ones being fixed at zero.

Thus far, we have not discussed the optimization over the $\tilde{w}$ terms (the weights). The weights must all be positive and sum to one. Williams and Maybeck [14] considered including them in the iterative optimization over all of the parameters. Schieferdecker and Huber [20] used an explicit unconstrained solution and Chen, Chang and Smith [3] used an explicit solution that had been constrained such that the weights must sum to unity, but which *did not rule out negative weights*. This is not merely a theoretical problem, since we have observed, through simulation, that negative weights can occur in practice when using an unconstrained approach. Note that the COWA optimizes only over the weights and nothing else in the second step of the algorithm. The full optimization over the weights given the means and covariances of the Gaussian components of the mixture is a *convex quadratic programming problem*, and (omitting the constant term from (7) and multiplying everything by $1/2$), may be formulated as

$$\min_{\tilde{\mathbf{w}}} \quad \frac{1}{2}\tilde{\mathbf{w}}^T H_1 \tilde{\mathbf{w}} - \mathbf{w}^T H_2 \tilde{\mathbf{w}} \tag{32}$$

$$\text{subject to} \quad \mathbf{1}^T \tilde{\mathbf{w}} = 1 \tag{33}$$

$$-\tilde{\mathbf{w}} \leq \mathbf{0} \tag{34}$$

where

$$\mathbf{1} \triangleq [1, 1, \ldots, 1]^T \qquad \mathbf{0} \triangleq [0, 0, \ldots, 0]^T \tag{35}$$

Because H_1 is a positive definite matrix[7], this is a convex quadratic programming problem and can thus be reduced to a linear programming problem and solved exactly in polynomial time [25], [27]. The inequality constraint prevents the problem from having an explicit solution.

All together, given the estimate of the parameters from the second step of the GMRC, one can perform Quasi-Newton optimization over the means and the nonzero parameters of the Cholesky decomposition of the covariance matrices, whereby for each hypothesis the globally optimal mixture weights can be found via quadratic programming.[8] In the COWA, the optimization is performed only over the weights.

IV. A Brute Force Algorithm

The first two steps of the GMRC, discussed in Section III, address the fact that an iterative optimization over the ISE is highly dependent upon the initial estimate used. Often the choice of which components of the original mixture to merge, as determined by Algorithms 2 or 3, is suboptimal and can be improved by k-means refinement, which still does not guarantee global optimality. In this section, we consider the implementation of a brute-force initialization algorithm that yields the globally optimal clustering for initialization.[9] The optimal algorithm provides a basis for comparison against other algorithms when run on problems where the brute force solution is feasible.

A large number of globally optimal algorithms for clustering exist, for example [28], but none currently use the ISE as the cost function.[10] Thus, though the algorithm presented here is slow, it is nonetheless optimal with respect to the ISE.

First, let us consider the complexity of implementing a brute-force clustering and reduction algorithm. The total number of possible ways of grouping components of an N-component mixture to reduce it to an L component mixture is given by $S(N, L)$[11], a Stirling number of the second kind [30] having equation

$$S(N, L) = \frac{1}{L!}\sum_{i=0}^{L} (-1)^i \binom{L}{i} (L - i)^N \tag{36}$$

[7]**Proof:** By definition [25], a real matrix, Q is positive definite if $\mathbf{a}^T Q \mathbf{a} > 0 \, \forall \mathbf{a} \neq \mathbf{0}$. For an arbitrary $L \times 1$ real vector $\mathbf{a}$ we can write:

$$\mathbf{a}^T H_1 \mathbf{a} = \mathbf{a}^T \left(\int_{\mathbf{x}} \tilde{\mathbf{N}}(x)\tilde{\mathbf{N}}(x)^T \, d\mathbf{x}\right)\mathbf{a} = \int_{\mathbf{x}} \mathbf{a}^T \tilde{\mathbf{N}}(x)\tilde{\mathbf{N}}(x)^T \mathbf{a} \, d\mathbf{x} = \int_{\mathbf{x}} \left(\mathbf{a}^T \tilde{\mathbf{N}}(x)\right)^2 \, d\mathbf{x} \geq 0$$

We know that all of the elements of $\tilde{\mathbf{N}}(x)$ must be nonzero for finite $\mathbf{x}$. Consequently, H_1 is a positive definite matrix.

[8]In MATLAB, one can use `fminunc` with the gradient option turned on for the overall optimization and `quadprog` for the weight optimization.

[9]In other words, we look at all possible ways of merging the data (using (18)–(20)) and choose the optimal solution.

[10]An extended version of this paper is in preparation that will discuss a dynamic programming solution, similar to that in [29], with branch-and-bound on a slightly modified cost function, which is significantly faster.

[11]$\left\{ {n \atop L} \right\}$ is also used to represent a Stirling number of the second kind.

$$S(N,L)=\sum_{i_1=1}^{\lfloor \frac{N}{L} \rfloor}\sum_{i_2=i_1}^{\lfloor \frac{N-i_1}{L-1} \rfloor}\sum_{i_3=i_2}^{\lfloor \frac{N-i_1-i_2}{k-2} \rfloor}\cdots\sum_{i_{L-1}=i_{L-2}}^{\left\lfloor \frac{N-\sum_{j=1}^{L-2} i_j}{2} \right\rfloor}\frac{\prod_{j=1}^{k-1}\binom{N-\sum_{n=1}^{j-1} i_n}{i_j}}{c_R\left[i_1,i_2,\ldots,i_{L-1},N-\sum_{j=1}^{L-1} i_j\right]} \tag{39}$$

While the evaluation of the ISE for a given reduced mixture is straightforward, to "wrap" that within an exhaustive routine is, perhaps surprisingly, not. The concern is to structure an automated procedure (presumably a set of nested loops) to visit each feasible reduced mixture exactly once, and that structure is a key contribution of this paper.

To develop this structure, we offer an alternative derivation of (39), which counts all possible ways of putting N items into L unordered bins, such that no bin is ever empty. Since the order of the bins does not matter, we will define an arbitrary (ranked) ordering so as to avoid double counting. For example, if we had three bins and five items, assignments of items into bins having ordered sizes $(1,1,3)$ would be identical to assignments in bins having ordered sizes $(1,3,1)$ or $(3,1,1)$. Thus, we shall only count assignments such that bin i contains no more than the number of items in bin j if $i \geq j$. In other words, the number of items in the bins always increases or stays the same. Note that this does not eliminate double-counting if multiple bins have the same size, which we shall address later.

The first bin can contain anywhere from $i_1^{MIN} = 1$ to $i_1^{MAX} = \lfloor N/(L-1+1) \rfloor$ items, since all subsequent bins must be at least the same size or larger. Similarly, the second bin may contain any number of items ranging from what is in the first bin, $i_2^{MIN} = i_1$ to $i_2^{MAX} = \lfloor (N-i_1)/(L-2+1) \rfloor$. The L^{th} bin must contain the proper number of items such that the sum of the number of items in each bin is equal to N. In general, we find that the N^{th} bin can contain:

$$i_n^{MIN} = \begin{cases} 1 & \text{if } n = 1 \\ i_{n-1} & \text{if } 1 < n < L \\ N - \sum_{j=1}^{n-1} i_j & \text{if } n = L. \end{cases} \tag{37}$$

$$i_n^{MAX} = \left\lfloor \frac{N - \sum_{j=1}^{n-1} i_j}{L-n+1} \right\rfloor \tag{38}$$

Note that if $n = L$, $i_{MIN} = i_{MAX}$.

The total number of possible clusterings using this counting method is given in Equation (39) at the top of the page. The numerator in the equation multiplies out the possible ways of putting the N items into L bins having sizes determined by a particular set of i^{s} representing the sizes of the bins, with the final bin holding $\left(N - \sum_{j=1}^{L-1} i_j\right)$ items. The denominator divides out repeats in the counting.

Suppose that we have r bins of size b. When we go through all possible ways of putting things into those bins we will repeat each pattern $b!$ times. For example, suppose that $b = 2$ and $r = 2$. In one combination, we might assign items 1 and 2 to the first bin and 3 and 4 to the second bin. In another combination, we might assign 3 and 4 to the first bin and 1 and 2 to the second bin. However, since we want to count the bins in an unordered manner, both of those possibilities are identical. Thus, the total number of unique assignments must be divided by $2! = 2$.

The function c_R determines the number of repeated partition sizes among the i terms. It then takes the product of the factorials of the number of repeats. For example, some of the values for ten elements being broken into five groups would be

$$c_R(1,1,1,1,6) = 4!1! \tag{40}$$

$$c_R(1,1,2,3,3) = 2!1!2! \tag{41}$$

$$c_R(2,2,2,2,2) = 5! \tag{42}$$

From (39), we can get an insight into how a brute force clustering algorithm may be implemented. First, we can generate instances of how many observations are in each cluster by using loops analogously to the sums in (39). Generating all possible combinations of what goes into the bins may be done sequentially using loops bin by bin. That is, if the first bin contains l_1 items, we will enumerate all possible combinations of l_2 items. For each combination, we will remove those items from the set of N things that we can choose from, and do the same thing for the second bin and so on. For bin i, we will go through all possible combinations of l_i of the remaining items, remove those items from the set and continue to the next bin. However, things become more difficult when we have multiple bins containing the same number of items.

When we come to a stretch of $B > 1$ bins containing the same number of elements, we can no longer use the same recursion as above lest we waste effort evaluating these $B!$ identical clusterings. Thus, suppose that S is the set of elements remaining from which we can draw. We have a sequence of $B > 1$ bins that each containing l_b items. We shall go through the combinations of what goes into the b bins, plus the subsequent bins via the following recursion:

The Recursion for Repeated Bins

1) S is the set of items that have not been assigned to bins before the current bin. Generate all possible combinations, C_S, of l_b items from the set S.
2) Sequentially go through which combination goes into the first bin of the repeats.
3) To go to the next bin, remove the current combination and all combinations previously visited by this bin from C_S and remove all combinations containing elements in the current combination from C_S and remove the chosen elements from S and pass the modified C_S and S to the next level, which will repeat from step 2 on the reduced set.

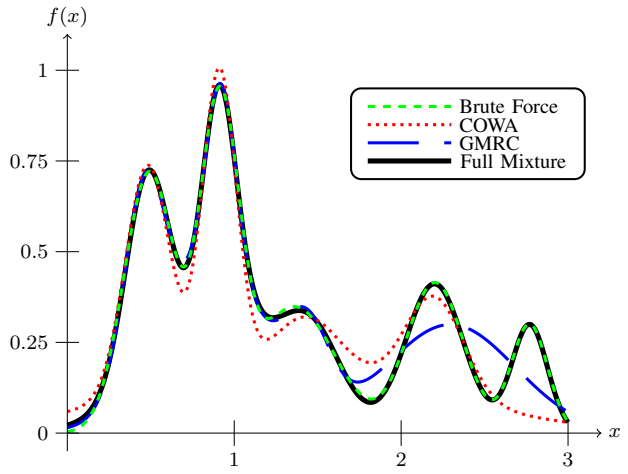

Figure 1. A comparison of the methods using the three primary techniques discussed for reducing a 10 component Gaussian mixture to 5 components. Note that the line for the brute force approach overlaps that of the optimal solution most of the time.

4) Once the final bin containing l_b items has been filled and S updated, if the current bin is not the final bin, continue assigning things to the subsequent bins (which will each hold fewer than I_b items) either using the method for when there are no repeats or the method for repeats, as appropriate.

In order to ensure that we get the best possible estimate for comparison, we will use the iterative optimization of Section III-C to refine this brute-force estimate when simulating.

V. Simulations

	ISE	NISE	Execution Time (sec)
COWA	0.1165	0.1080	0.0390
Runnalls	0.0834	0.0814	0.0048
GMRC	0.0482	0.0432	53.4499
Brute Force	0.0359	0.0309	622.0235

Table I
THE PERFORMANCE AND EXECUTION TIMES OF THE FOUR METHODS IN THE 4-DIMENSIONAL RANDOMIZED REDUCTION SCENARIO REDUCING A 10 COMPONENT MIXTURE TO 5 COMPONENTS. THE RESULTS ARE AVERAGED OVER 500 MONTE CARLO RUNS.

We compared the performance of the modified GMRC and COWA algorithms described in Section III against the brute force approach of Section IV and against simply using Runnalls' algorithm (the first step of the GMRC). The simulation was run reducing randomly generated $D = 4$ dimensional $N = 10$ component Gaussian mixtures to $L = 5$ component mixtures, meaning that the brute-force approach had to calculate the cost of $42,525$ possible reductions each time. The COWA algorithm was run with $\gamma = \infty$ and $\epsilon = 0$, forcing the algorithm to reduce the mixture to the minimum number of components each time. The components of the means were chosen uniformly between 0 and 3. The covariances of the mixture components were chosen from a Wishart distribution [31] with scale matrix $\frac{1}{50}\mathbf{I}$ and 5 degrees of freedom. The mean covariance for each component was thus $\frac{1}{10}\mathbf{I}$. The weights for the components were chosen from a symmetric Dirichlet distribution [32] with parameter $\alpha = 1$. The execution times of the algorithms and their performance in terms of the normalized integral squared difference, as defined in Equation (17), which ranges from zero to one (smaller numbers are better) is given in Table I. The simulation was run on a 2.4GHz computer running the Mac OS X version of MATLAB. From Table I, we can see that the COWA lags behind the faster, lower-complexity algorithm by Runnalls. The improvement over Runnalls' algorithm by the GMRC comes at a steep computational cost. All algorithms have room for improvement in terms of ISE and ISE compared to the slow, brute-force algorithm.

To better visualize the performance of the reduction algorithms, we considered a scalar scenario using the following parameters:

$$\begin{aligned} w =&\{0.03, 0.18, 0.12, 0.19, 0.02, 0.16, 0.06, 0.1, 0.08, 0.06\} \\ \mu =&\{1.45, 2.20, 0.67, 0.48, 1.49, 0.91, 1.01, 1.42, 2.77, 0.89\} \\ P =&\{0.0487, 0.0305, 0.1171, 0.0174, 0.0295, \\ &\quad 0.0102, 0.0323, 0.0380, 0.0115, 0.0679\} \end{aligned}$$

The 10-component mixture was reduced to five components using the COWA, the GMRC and the brute-force algorithm. The result is shown in Figure 1. We can see that the reduced mixture using the brute-force reduction approach is the only method that avoids large deviations from the original mixture.

However, to what extent does this matter in a practical algorithm? To address this question, we implemented Salmond's [6] Gaussian mixture reduction-based tracking algorithm using the COWA and the GMRC. We considered tracking a single target in 2D using the standard motion model used with the Kalman filter, namely

$$x_t(k+1) = F(k)x_t(k) + v_t(k) \quad (43)$$
$$z_t(k) = H_t(k)x_t(k) + w_t(k) \quad (44)$$

where x is the state, z is a Cartesian position measurement and v_t and w_t are independent Gaussian white noise. We used a sampling interval of $\tau = 1s$. We maintained $L = 3$ hypotheses across time-steps. We used the discretized continuous time white noise acceleration model from [9] with process noise parameter $q = 2$. The target started at the origin having an initial velocity of $10\,\mathrm{m/s}$ in both Cartesian components. The measurement covariance was a diagonal matrix with 60 on both of the diagonal elements. The probability of detecting the target was 70%. Rectangular gates enclosing the 99.9% confidence region (which is elliptical) were used about each hypothesized target state. Clutter was generated in the gate according to a Poisson distribution having density $\lambda = 5.6 \times 10^{-5}$. The number of clutter points in each gate varied, but were often around $1-5$. The initial state was determined using two measurements from the true target. The trackers were run for 1000 Monte Carlo runs for a maximum of 50-time steps.

Table II shows the percentage of lost tracks as a function of the tracker used as well as the RMSE of the tracks that were not lost. Tracks were declared lost if the true target location

	RMSE	% Lost Tracks
COWA	9.11	25.35
Runnalls	7.98	3.15
GMRC	8.26	3.25

Table II
THE PERCENTAGE OF LOST TRACKS OVER A 50-STEP PERIOD, AS WELL AS THE RMSE OF THE TRACKS THAT WEREN'T LOST, AS A FUNCTION OF THE MIXTURE REDUCTION ALGORITHM USED IN IMPLEMENTATIONS OF SALMOND'S TRACKER WITH $L = 3$ HYPOTHESES KEPT.

was not in any of the gates at a particular time-step or if no hypotheses were left. Hypotheses were declared lost and were subsequently pruned if the gate for any of them exceeded 500 meters in either dimension. We can see that the quality of the reduction algorithm is important, with Runnalls' algorithm and the GMRC having significantly fewer lost tracks than the COWA. However, there is no noticeable improvement between the Runnalls' algorithm and the GMRC.

VI. CONCLUSIONS

In Section III-C, it was noted that the accepted optimization of the weight terms in the GMRC and COWA algorithms could result in the reduced Gaussian mixture being invalid. We have made an appropriate modification to address this. We also combined concepts from other papers, generalizing the GMRC to vector distributions. To form a basis for comparison with other algorithms, we derived a brute-force approach. From the simulations using random Gaussian mixtures, we can see that the GMRC is the best algorithm (in terms of ISE) in comparison to the brute force algorithm. Indeed, the COWA was inferior to Runnalls algorithm both in performance as well as execution time. However, the improvement in running the full GMRC versus simply running the first step (Runnalls' algorithm) is nonexistent when looking at a tracking scenario, suggesting that a naked Runnalls' algorithm would be the most practical Gaussian mixture reduction algorithm to use in a tracker thus far.

REFERENCES

[1] B. Krukowski and J. Dauwels, "Message passing decoding of lattices using Gaussian mixtures," in *Proceedings of the IEEE International Symposium on Information Theory*, Toronto, ON, Canada, Jul. 2008, pp. 2489–2493.

[2] A. Nikseresht and M. Gelgon, "Gossip-based computation of a Gaussian mixture model for distributed multimedia indexing," *IEEE Trans. Multimedia*, vol. 10, no. 3, pp. 385–392, Apr. 2008.

[3] H. Chen, K. C. Chang, and C. Smith, "Constraint optimized weight adaptation for Gaussian mixture reduction," in *Proceedings of SPIE: Signal Processing, Sensor Fusion, and Target Recognition XIX*, vol. 7697, Orlando, FL, Apr. 2010, pp. 76 970N–1–76 970N–10.

[4] P. Bruneau, M. Gelgon, and F. Picarougne, "Parsimonious reduction of Gaussian mixture models with a variational-Bayes approach," *Pattern Recognition*, vol. 43, no. 3, pp. 850–858, Mar. 2010.

[5] S. S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 19, no. 1, pp. 5–18, Jan. 2004.

[6] D. J. Salmond, "Mixture reduction algorithms for point and extended object tracking in clutter," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 45, no. 2, pp. 667–686, Apr. 2009.

[7] ——, "Mixture reduction algorithms for target tracking in clutter," in *Proceedings of SPIE: Signal and Data Processing of Small Targets Conference*, vol. 1305, Oct. 1990, pp. 434–445.

[8] L. Y. Pao, "Multisensor mixture reduction algorithms for tracking," *Journal of Guidance, Control and Dynamics*, vol. 17, no. 6, pp. 1205–1211, Nov.–Dec. 1994.

[9] Y. Bar-Shalom, P. K. Willett, and X. Tian, *Tracking and Data Fusion*. YBS Publishing, 2011.

[10] D. J. Petrucci, "Gaussian mixture reduction for Bayesian target tracking in clutter," Master's thesis, Air Force Institute of Technology, Dec. 2005. [Online]. Available: http://www.dtic.mil/srch/doc?collection=t3&id=ADA443588

[11] D. W. Scott, "Parameter modeling by minimum L_2 error," Department of Statistics, Rice University, Tech. Rep. 98-3, Feb. 1999. [Online]. Available: http://www.stat.rice.edu/~scottdw/ftp/l2e.r1.ps

[12] D. W. Scott and W. F. Szewczyk, "From kernels to mixtures," *Technometrics*, vol. 43, no. 3, pp. 323–335, Aug. 2001.

[13] J. E. Harmse, "Reduction of Gaussian mixture models by maximum similarity," *Journal of Nonparametric Statistics*, vol. 22, no. 6, pp. 703–709, Dec. 2009.

[14] J. L. Williams and P. S. Maybeck, "Cost-function-based hypothesis control techniques for multiple hypothesis tracking," *Mathematical and Computer Modelling*, vol. 43, no. 9–10, pp. 976–989, May 2006.

[15] J. L. Williams, "Gaussian mixture reduction for tracking multiple maneuvering targets in clutter," Master's thesis, Air Force Institute of Technology, Mar. 2003. [Online]. Available: http://www.dtic.mil/srch/doc?collection=t3&id=ADA415317

[16] M. Rosenblatt, "Remarks on some nonparametric estimates of a density function," *The Annals of Mathematical Statistics*, vol. 27, no. 3, pp. 832–837, Sep. 1956.

[17] M. West, "Approximate posterior distributions by mixture," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 55, no. 2, pp. 409–422, 1993.

[18] M. F. Huber and U. D. Hanebeck, "Progressive Gaussian mixture reduction," in *Proceedings of the 11th International Conference on information Fusion*, Cologne, Germany, Jan. 2008, pp. 1–8.

[19] O. C. Schrempf, O. Feiermann, and U. D. Hanebeck, "Optimal mixture approximation of the product of mixtures," in *Proceedings of the Eighth International Conference on information Fusion*, Philadelphia, PA, Jul. 2005, pp. 85–92.

[20] D. Schieferdecker and M. F. Huber, "Gaussian mixture reduction via clustering," in *Proceedings of the 11th International Conference on information Fusion*, Seattle, WA, Jul. 2009, pp. 1536–1543.

[21] A. R. Runnalls, "Kullback-Leibler approach to Gaussian mixture reduction," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 43, no. 3, pp. 989–999, Jul. 2007.

[22] K. C. Chang and W. Sun, "Scalable fusion with mixture distributions in sensor networks," in *Proceedings of the 11th International Conference on Control, Automation, Robotics, and Vision*, Singapore, Dec. 2010, pp. 1251–1256.

[23] C. Bishop, *Pattern Recognition and Machine Learning*. Cambridge, U.K.: Springer, 2006.

[24] T. Kanungo, D. M. Mount, D. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 881–892, Jul. 2002.

[25] D. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, Massachusetts: Athena Science, 2003.

[26] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore: Johns Hopkins University Press, Oct. 1996.

[27] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*, 1st ed. Belmont, Massachusetts: Athena Science, Feb. 1997.

[28] D. Aloise, "Exact algorithms for minimum sum-of-squares clustering," Ph.D. dissertation, École Polytechnique de Montréal, Jun. 2009. [Online]. Available: http://www.gerad.ca/~aloise/These.pdf

[29] R. E. Jensen, "A dynamic programming algorithm for cluster analysis," *Operations Research*, vol. 17, no. 6, pp. 1034–1057, Nov.–Dec. 1969.

[30] E. W. Weisstein, *CRC Concise Encyclopedia of Mathematics*. Boca Raton: CRC Press, 1999, p. 1741.

[31] M. L. Eaton, *Multivariate Statistics: A Vector Space Approach*, ser. Institute of Mathematical Statistics Lecture Notes-Monograph Series. Beachwood, Ohio: Institute of Mathematical Statistics, 2007, vol. 53, ch. 5 & 8.

[32] B. A. Frigyik, A. Kapila, and M. R. Gupta, "Introduction to the Dirichlet distribution and related processes," University of Washington, Tech. Rep. UWEETR-2010-0006, Dec. 2010. [Online]. Available: https://www.ee.washington.edu/techsite/papers/refer/UWEETR-2010-0006.html