ℓ_0-Regularized Sparsity for Probabilistic Mixture Models

Dzung T. Phan * Tsuyoshi Idé *

Abstract

This paper revisits a classical task of learning probabilistic mixture models. Our major goal is to sparsely learn the mixture weights to automatically determine the right number of clusters. The key idea is to use a novel Bernoulli prior on the mixture weights in a Bayesian learning framework, and formalize the task of determining the mixture weights as an ℓ_0-regularized optimization problem. By leveraging a specific mathematical structure, we derive a quadratic time algorithm for efficiently solving the non-convex ℓ_0-based problem. In experiments, we evaluate the performance of our proposed approach over existing methods in recovery capability and anomaly detection for synthetic as well as real-world data sets.

1 Introduction

Probabilistic mixture models provide a powerful approach to modeling real-world complex distributions. Thanks to their simplicity and interpretability, they have been used in numerous applications. When using mixture models, one of the most important decision points is how to determine the number of mixture components. In spite of the seeming simplicity, correctly inferring the number of clusters is known to be challenging because of the singularity of the model.

One standard solution to this problem is to use infinite mixture models based on the non-parametric Bayesian framework [14]. The idea is to start with a model with an as large number of clusters as possible and let the algorithm choose active clusters. In a sense, such an approach can be viewed as finding *sparse mixture weights* over a large number of candidate clusters. The infinite mixture methods require rather complicated inference procedures via Monte Carlo sampling [10, 11] or variational inference [18, 3, 16], which undermine the original benefit of simplicity of mixture models.

The other interesting but less known approach for sparse mixture weights is to leverage an automatic relevance determination (ARD) mechanism in Bayesian learning [5]. As described in the next section in detail, it treats the mixture weights as model parameters of a latent variable. Then it determines the mixture weights by maximizing the marginalized likelihood (Eq. (2.3)). Empirically it is known that this simple approach produces a sparse numerical solution. However, rather strangely, the optimization problem to determine the mixture weights does not mathematically have a sparse solution in the standard sense because of the logarithmic terms (Eq. (2.4)). In a manner, a sparse solution is obtained only as a numerical artifact. Furthermore, there is no hyper-parameter to indirectly or directly control the number of components; hence we cannot regulate sparsity level. As a result, the approach may not generate sufficiently sparse mixture models.

Motivated by these observations, we propose a new approach to sparsify mixture weights of finite mixture models by pruning irrelevant mixture components. In particular, we introduce a novel Bernoulli prior to the mixture model and formalize the task of finding mixture weight as an ℓ_0-regularized optimization problem. Although directly handling the ℓ_0 norm is generally known to be challenging, due to the strongly-structured nature of the problem, we can develop an efficient quadratic time algorithm. To the best of our knowledge, this is the first work to formalize the ARD mechanism of probabilistic mixture models using ℓ_0 regularization and derive its practical algorithm.

2 Background: probabilistic mixture models

We start with recapturing the basic setup of probabilistic mixture models. For more details, readers may refer to Section 10.2 of [2]. Consider a K-component mixture model for an observable variable $\boldsymbol{x} \in \mathbb{R}^M$ as

$$P(\boldsymbol{x} \mid \boldsymbol{z}, \boldsymbol{\Theta}) = \prod_{k=1}^{K} p(\boldsymbol{x} \mid \boldsymbol{\theta}_k)^{z_k}, \qquad p(\boldsymbol{z} \mid \boldsymbol{\pi}) = \prod_{k=1}^{K} \pi_k^{z_k}, \quad \sum_{k=1}^{K} \pi_k = 1, \tag{2.1}$$

where $z_k \in \{0, 1\}$ is the indicator variable of cluster assignment, $\boldsymbol{\Theta} \equiv \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K\}$ is a collection of model parameters of each component, and $\boldsymbol{\pi}$ is the vector of *mixture weights* to be determined from data. We assume that we are given N samples for $\boldsymbol{x}$ as $\mathcal{D} \equiv \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\}$. The complete likelihood of this model

*IBM Research, T. J. Watson Research Center, New York, USA, Email: {phandu,tide}@us.ibm.com.

is then given by

$$P(\mathcal{D}, \mathsf{Z}, \boldsymbol{\Theta} \mid \boldsymbol{\pi}) = P(\boldsymbol{\Theta}) \prod_{n=1}^{N} P(\boldsymbol{x}^{(n)} \mid \boldsymbol{z}^{(n)}, \boldsymbol{\Theta}) p(\boldsymbol{z}^{(n)} \mid \boldsymbol{\pi}) \tag{2.2}$$

where $P(\boldsymbol{\Theta})$ is a prior distribution for $\boldsymbol{\Theta}$ and $\mathsf{Z} \equiv \{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(N)}\}$, $\boldsymbol{z}^{(n)}$ is the indicator variable assigned to the n-th sample.

The goal of Bayesian inference for this model is to find the posterior distributions of Z and $\boldsymbol{\Theta}$. The variational Bayes (VB) method is a useful framework to approximately find the posterior distribution under the assumption that they are factorized as $Q(\mathsf{Z}, \boldsymbol{\Theta}) = u(\mathsf{Z})v(\boldsymbol{\Theta})$. By minimizing the Kullback–Leibler divergence between the assumed factorized form and the true posterior, one can derive update equations as

$$\ln u(\mathsf{Z}) = \text{c.} + \langle \ln P(\mathcal{D}, \mathsf{Z}, \boldsymbol{\Theta} \mid \boldsymbol{\pi}) \rangle_{\boldsymbol{\Theta}},$$
$$\ln v(\boldsymbol{\Theta}) = \text{c.} + \langle \ln P(\mathcal{D}, \mathsf{Z}, \boldsymbol{\Theta} \mid \boldsymbol{\pi}) \rangle_{\mathsf{Z}}$$

for a given $\boldsymbol{\pi}$, where $\langle \cdot \rangle_{\boldsymbol{\Theta}}$ and $\langle \cdot \rangle_{\mathsf{Z}}$ represent expectation w.r.t. $v(\boldsymbol{\Theta})$ and $u(\mathsf{Z})$, respectively. The authors of [5] numerically showed that by combining these equations with a point-estimation equation

$$\boldsymbol{\pi} = \arg\max_{\boldsymbol{\pi}} \langle \ln P(\mathcal{D}, \mathsf{Z}, \boldsymbol{\Theta} \mid \boldsymbol{\pi}) \rangle_{\boldsymbol{\Theta}, \mathsf{Z}}, \tag{2.3}$$

one can achieve sparsity over $\boldsymbol{\pi}$ through the ARD mechanism, allowing automated determination of the number of mixture components. Using Eqs. (2.1) and (2.2), we can see that (2.3) is reduced to

$$\max_{\boldsymbol{\pi}} \sum_{k=1}^{K} r_k \ln \pi_k \quad \text{subject to} \quad \sum_{k=1}^{K} \pi_k = 1, \tag{2.4}$$

where $r_k \equiv \sum_{n=1}^{N} \langle z_k^{(n)} \rangle_{\mathsf{Z}}$. This is the main problem we are interested in.

Sparsity for mixture weights related to the update (2.4) has been rarely addressed explicitly in the literature, partly because π_k cannot be mathematically zero. Although it is *empirically* known that the solution can achieve sparsity in many cases when combined with a heuristically determined threshold in numerical calculations to decide on zero components in $\{\pi_k\}$. We note that there is no mathematical guarantee for yielding a sparse solution from (2.4) since no sparsity constraints are explicitly imposed. As a consequence, it may not generate sufficiently sparse estimates. Unfortunately, the common idea of using ℓ_1 regularizer does not work in this case because the value of $\|\boldsymbol{\pi}\|_1$ is fixed (=1). We have a similar issue with alternative hierarchical Bayesian formulations such as the stick-breaking process [18, 3, 8, 16] and Markov chain Monte-Carlo sampling method [10, 11]. Our main motivation is to explicitly introduce a sparsity-enforcing mechanism over the mixture weights by placing a Bernoulli prior on $\boldsymbol{\pi}$.

3 Convex mixed-integer programming formulation for mixture weights

To explicitly deriving a sparsity promoting procedure, we place a hyper prior on $\boldsymbol{\pi}$. Especially, we use the Bernoulli prior

$$p(\boldsymbol{\pi}) = \gamma^{\|\boldsymbol{\pi}\|_0} (1-\gamma)^{K - \|\boldsymbol{\pi}\|_0} \tag{3.5}$$

for the mixture weights, where $\|\cdot\|_0$ is the ℓ_0-norm (the number of nonzeros). We can recast Eq. (2.3) as

$$\begin{aligned} &\max_{\boldsymbol{\pi}} \quad \textstyle\sum_{k=1}^{K} a_k \ln(\pi_k) - \tau \|\boldsymbol{\pi}\|_0, \\ &\text{s.t.} \quad \textstyle\sum_{k=1}^{K} \pi_k = 1, \quad \pi_k \geq 0, \end{aligned} \tag{3.6}$$

where we introduce $a_k \equiv \frac{r_k}{N}$ for simplicity and $\tau > 0$ is a regularization parameter.

Obviously, the problem (2.4) is recovered when $\tau = 0$. Although solving a general nonconvex ℓ_0-norm optimization problem is computationally challenging [19], we can show that the problem can be solved efficiently in our particular setting. First, we reformulate Eq. (3.6) as a convex mixed-integer programming (MIP)

$$\begin{aligned} \min_{\boldsymbol{\pi}, \boldsymbol{y}} \quad & -\textstyle\sum_{k=1}^{K} a_k \ln(\pi_k) + \tau \sum_{k=1}^{K} y_k \\ \text{s.t.} \quad & \textstyle\sum_{k=1}^{K} \pi_k = 1, \pi_k \geq 0, \\ & y_k \geq \pi_k, y_k \in \{0,1\}, k = 1, \ldots, K. \end{aligned} \tag{3.7}$$

The equivalence between (3.6) and (3.7) comes from the fact that $||\boldsymbol{\pi}||_0 = \sum_{k=1}^{K} y_k$ for (3.7). In view of the logarithmic term $\ln(\pi_k)$, even explicitly enforcing the sparsity promoting ℓ_0 norm, all the solutions π_k of (2.4) and (3.7) will be nonzero. This means that we cannot get a sparse solution for $\boldsymbol{\pi}$ in the standard sense. These facts bring us to a new notion of *asymptotic* sparsification: we wish to have

$$|\pi_k| \leq \epsilon$$

for many k's, where ϵ is a very small number. Putting formally,

DEFINITION 1. *For a given small $\epsilon > 0$, a vector $\boldsymbol{x}$ is called an ϵ-sparse solution if many elements satisfy $|x_i| \leq \epsilon$.*

To find an ϵ-sparse solution, we propose to solve a modified version of Eq. (3.7) by perturbing $y_k \geq \pi_k$:

$$\begin{aligned} \min_{\boldsymbol{\pi}, \boldsymbol{y}} \quad & f(\boldsymbol{\pi}, \boldsymbol{y}) \equiv -\textstyle\sum_{k=1}^{K} a_k \ln(\pi_k) + \tau \sum_{k=1}^{K} y_k \\ \text{s.t.} \quad & \textstyle\sum_{k=1}^{K} \pi_k = 1, \pi_k \geq 0, \\ & y_k \geq \pi_k - \epsilon, y_k \in \{0,1\}, k = 1, \ldots, K. \end{aligned} \tag{3.8}$$

By using (3.8), we encourage to retain only important components. We will see later that the second term $\sum_{k=1}^{K} y_k$ equals to the number of elements π_k satisfying $\pi_k > \epsilon$. The following theorem guarantees that we can achieve a sparse solution over mixture components in the VB formulation:

THEOREM 3.1. *The following statements hold for Eq.* (3.8)*:*

(i) *It is a convex mixed-integer programming with a bounded polyhedron feasible set.*

(ii) *For a suitable selection of the regularization parameter τ, the problem* (3.8) *generates an ϵ-sparse solution.*

(iii) *There exist small enough positive numbers τ and ϵ such that if $(\boldsymbol{\pi}, \boldsymbol{y})$ is a solution of* (3.8) *then $\boldsymbol{\pi}$ is a minimizer of* (2.4)*.*

We note that (3.8) is only a relaxed version of (3.6), they are not equivalent. Thus, the classical idea of using the ℓ_0 regularization to control sparsity cannot be directly applied in our setting. We introduced a new ϵ-sparsity notation and a mixed integer programming. Hence, it is necessary to formally point out how the parameter τ control ϵ-sparsity. Furthermore, our result in Theorem 3.1 (iii) is stronger than the conventional one since we claim it in an absolute sense (hold true for a fixed small τ) as opposed to the known asymptotic sense (hold true when τ tends zero).

Proof. (i) We can see that the Hessian of the objective is $\boldsymbol{H} = \operatorname{diag}\left(\frac{a_1}{\pi_1^2}, \ldots, \frac{a_K}{\pi_K^2}, 0, \ldots, 0\right)$, where $\operatorname{diag}(\boldsymbol{x})$ is an $2K$ by $2K$ diagonal matrix with i-th diagonal element x_i. From the definition of a_k in Section 2, it follows that a_k is strictly positive for every k. Hence $\boldsymbol{H}$ is a positive semi-definite matrix, which implies that the objective function is convex. It is easy to see that all decision variables are bounded in $[0, 1]$, and every constraint is linear. Thus, the feasible set is a polytope.

(ii) We claim that the sum $\sum_{k=1}^{K} y_k$ is equal to the number of π_k satisfying $\pi_k > \epsilon$. Since $y_k \geq \pi_k - \epsilon$ and $y_k \in \{0, 1\}$, if $\pi_k > \epsilon$, then $y_k = 1$. If $\pi_k \leq \epsilon$, because we are solving a minimization problem, then one must have $y_k = 0$. Hence the necessary and sufficient condition for $y_k = 1$ is $\pi_k > \epsilon$. We can choose a value for τ to balance the first term in the objective and the number of nonzero y_i, which proves the statement.

(iii) Denote $\hat{\boldsymbol{\pi}}$ and $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{y}})$ by the optimal solutions of problems (2.4) and (3.8), respectively. Define $\hat{f}$ and $\bar{f}$ as the optimal function values of (2.4) and (3.8), respectively. When $\tau = 0$, the first-order optimality conditions for Eq. (2.4) readily give the solution $\hat{\pi}_k \propto a_k$, leading to $\hat{\pi}_k = \frac{a_k}{\sum_{l=1}^{K} a_l} = a_k$ as a unique solution of (2.4). We define

$$\epsilon = 0.5 \min\{\, a_k \mid \; k = 1, \ldots, K\}\,. \tag{3.9}$$

Since a_k are strictly positive for every k, then $\epsilon > 0$. Hence, we have $\hat{\pi}_k > \epsilon$ because of the definition of ϵ. Denote f_ϵ by the optimal function value of the following

$$\begin{aligned} f_\epsilon = \min_{\boldsymbol{\pi}} \quad & -\textstyle\sum_{k=1}^{K} a_k \ln(\pi_k) \\ \text{s.t.} \quad & \textstyle\sum_{k=1}^{K} \pi_k = 1 \\ & \pi_1 \leq \epsilon. \end{aligned} \tag{3.10}$$

When τ is chosen to be sufficiently small, $\tau \sum_{k=1}^{K} y_k$ is dominated by the first term in (3.8). Precisely, we can select

$$\tau = \frac{f_\epsilon - \hat{f}}{2K}. \tag{3.11}$$

Since both (2.4) and (3.10) are strongly convex problems and the feasible set of (3.10) is contained in the feasible set of (2.4), but it does not include the unique minimizer of (2.4) (because $\hat{\pi}_k > \epsilon$ for (2.4)), we have $f_\epsilon > \hat{f}$. Hence it follows $\tau > 0$. We claim that if ϵ and τ are defined as (3.9) and (3.11), then $\sum_{k=1}^{K} \bar{y}_k = K$ for (3.8), i.e., $\bar{y}_k = 1$ for every k . We proceed by contradiction. Assume that $\sum_{k=1}^{K} \bar{y}_k < K$. Hence, it implies $\bar{\pi}_1 \leq \epsilon$. We conclude that $\bar{\pi}$ is feasible to (3.10). We have

$$\begin{aligned} f(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{y}}) = \bar{f} \quad & \geq f_\epsilon + \tau \sum_{k=1}^{K} \bar{y}_k \\ & = \hat{f} + \tau(2K + \textstyle\sum_{k=1}^{K} \bar{y}_k) \\ & > \hat{f} + \tau K \\ & = f(\hat{\boldsymbol{\pi}}, \check{\boldsymbol{y}}), \end{aligned}$$

where $\check{\boldsymbol{y}} = \mathbf{1}$. This is a contradiction since $(\hat{\boldsymbol{\pi}}, \check{\boldsymbol{y}})$ is also a feasible solution of (3.8). When $\bar{y}_k = 1$ for all k, the problem (3.8) is reduced to (2.4). Thus it implies that $\bar{\boldsymbol{\pi}}$ is an optimal solution of (2.4) for small ϵ and τ. □

4 Efficient algorithm for the ϵ-sparse problem

In general, solving a MIP is NP-hard since it involves exhaustive combinatorial search. However, we can provide a semi-closed form solution for the problem (3.8). Our strategy is relatively simple. We find a solution of (3.8) for each discrete fixed value of $\sum_{k=1}^{K} y_k$, and select the best one from them.

Without loss of generality, we can assume that

$$0 < a_1 \leq a_2 \leq \ldots \leq a_K, \tag{4.12}$$

$$\text{and if } a_i = a_j \text{ for } i < j \text{ then } \pi_i \le \pi_j \tag{4.13}$$

for any solution $\boldsymbol{\pi}$ of (3.8). Note that $\sum_{k=1}^{K} a_k = 1$. Denote $\|\boldsymbol{y}\|_{\#}$ by the number of zero elements of $\boldsymbol{y}$. We can characterize the location of zero entries of $\boldsymbol{y}$ and the value $\|\boldsymbol{y}\|_{\#}$ in Lemma 4.1.

LEMMA 4.1. *Assume* $(\boldsymbol{\pi}, \boldsymbol{y})$ *is an optimal solution of* (3.8) *and the assumptions* (4.12) *and* (4.13) *are satisfied. We have:*

(i) *If* $\|\boldsymbol{y}\|_{\#} = m$ *then* $y_1 = \ldots = y_m = 0$ *and* $y_{m+1} = \ldots = y_K = 1$.

(ii) *It holds that* $\pi_k \le \pi_l$ *for every* $1 \le k < l \le K$.

Proof. (i) Assume that there exists some $y_k = 0$ with $m < k \le K$. It follows that there exists $y_l = 1$ where $1 \le l \le m$ because $\|\boldsymbol{y}\|_{\#} = m$. We have

$$\pi_k \le \epsilon < \pi_l. \tag{4.14}$$

Consider $(\overline{\boldsymbol{\pi}}, \overline{\boldsymbol{y}}) = (\pi_1, \ldots, \pi_k, \ldots, \pi_l, \ldots, x_K, y_1, \ldots, y_k, \ldots, y_l, \ldots, y_K)$. Then $(\overline{\boldsymbol{\pi}}, \overline{\boldsymbol{y}})$ is feasible to (3.8) and $\|\boldsymbol{y}\|_{\#} = \|\overline{\boldsymbol{y}}\|_{\#} = m$. One has

$$f(\boldsymbol{\pi}, \boldsymbol{y}) - f(\overline{\boldsymbol{\pi}}, \overline{\boldsymbol{y}}) = -(a_k - a_l)(\ln(\pi_k) - \ln(\pi_l)) \le 0. \tag{4.15}$$

Since $\pi_l > \pi_k$ and $a_l \ge a_k$, we have $a_l = a_k$. From the assumption (4.13), we get $\pi_l \le \pi_k$, which is a contradiction to (4.14).

(ii) If $a_l = a_k$ then $\pi_k \le \pi_l$ because of the assumption (4.13). If $a_l < a_k$, from (4.15) we can conclude that $\pi_k \le \pi_l$, which completes the proof. □

One of hidden parameters for the optimal solution of (3.8) is the number of zero elements y_k; that is the value $m = \|\boldsymbol{y}\|_{\#}$. We parameterize (3.8) using the parameter m. If m is known, by applying Lemma 4.1, solving (3.8) is equivalent to solving the convex continuous problem:

$$\begin{aligned} \min_{\boldsymbol{\pi}} \quad & -\textstyle\sum_{k=1}^{K} a_k \ln(\pi_k) \\ \text{s.t.} \quad & \textstyle\sum_{k=1}^{K} \pi_k = 1, \\ & \pi_k \le \epsilon, k = 1, \ldots, m, \\ & \pi_k > \epsilon, k = m+1, \ldots, K. \end{aligned} \tag{4.16}$$

We can use exhaustive search for $m = 0, \ldots, K-1$, then define $\boldsymbol{y}$ as

$$y_i = \begin{cases} 0, & \text{if } \pi_i \le \epsilon \\ 1, & \text{otherwise,} \end{cases}$$

and we get the value m giving the smallest $f(\boldsymbol{\pi}, \boldsymbol{y})$. The convex problem (4.16) can be solved in polynomial time by an interior-point method [4]; thus we can also solve (3.8) in polynomial time. We observe that the last constraints in (4.16) make it more complicated to be solved. We propose an alternative (4.17), which can be analytically solved for a fixed value m:

$$\begin{aligned} \min_{\boldsymbol{\pi}} \quad & -\textstyle\sum_{k=1}^{K} a_k \ln(\pi_k) \\ \text{s.t.} \quad & \textstyle\sum_{k=1}^{K} \pi_k = 1, \\ & \pi_k \le \epsilon, k = 1, \ldots, m. \end{aligned} \tag{4.17}$$

Now we show that Eq. (4.17) has a semi-closed form solution. Let us define

$$g(\boldsymbol{\pi}) = -\sum_{k=1}^{K} a_k \ln(\pi_k) + \tau |\{i : \pi_i > \epsilon\}|,$$

where $|\mathcal{S}|$ denotes the number of elements of the set $\mathcal{S}$. We need to search for m giving the smallest value for $g(\boldsymbol{\pi})$. For a given m, if $\overline{\boldsymbol{\pi}}$ and $\hat{\boldsymbol{\pi}}$ are solutions of (4.16) and (4.17) respectively, we have $g(\hat{\boldsymbol{\pi}}) \le g(\overline{\boldsymbol{\pi}})$. However, for an optimal value m of (3.8), it follows $g(\hat{\boldsymbol{\pi}}) = g(\overline{\boldsymbol{\pi}})$. The Lagrangian of (4.17) is

$$\begin{aligned} \mathcal{L}(\boldsymbol{\pi}, \boldsymbol{y}, \boldsymbol{\mu}, \lambda) = & -\sum_{k=1}^{K} a_k \ln(\pi_k) + \sum_{k \le m} \mu_k(\pi_k - \epsilon) \\ & + \lambda(\sum_{k=1}^{K} \pi_k - 1). \end{aligned}$$

The Karush-Kuhn-Tucker (KKT) conditions for (4.17) give

$$\begin{aligned} \frac{a_k}{\pi_k} &= \begin{cases} \lambda, & \text{if } k > m \\ \lambda + \mu_k, & \text{if } k \le m \end{cases} \\ \mu_k(\pi_k - \epsilon) &= 0, \quad k \le m \\ \mu_k &\ge 0, \quad k \le m. \end{aligned} \tag{4.18}$$

We are seeking a vector $\boldsymbol{\pi}$ which satisfies both the KKT conditions (4.18) and the feasibility for (4.17). The following lemma points out conditions for which some entries of $\boldsymbol{\pi}$ can be quickly determined.

LEMMA 4.2. *Assume* $\boldsymbol{\pi}$ *is an optimal solution of* (4.17) *for a given* $m = \|\boldsymbol{y}\|_{\#}$. *The following holds*

(i) *If* $a_k \ge \epsilon$ *and* $k \le m$ *then we have* $\pi_k = \epsilon$.

(ii) *If* $a_m \le \epsilon$ *or* $m = 0$ *then* $\boldsymbol{\pi} = \boldsymbol{a}$.

Proof. (i) We proceed by contradiction. Assume $\pi_k < \epsilon$ for some $k \le m$, then $\pi_k < \epsilon \le a_k$. From the complementary slackness in the KKT conditions (4.18), we have $\mu_k = 0$. It implies

$$\frac{a_k}{\pi_k} = \lambda > 1. \tag{4.19}$$

Since $\mu_k \geq 0$ for every $k \leq m$, we have $\frac{a_k}{\pi_k} \geq \lambda$ for all $1 \leq k \leq K$. It follows

$$\frac{a_k}{\lambda} \geq \pi_k.$$

Summing up the above inequality gives

$$\frac{\sum_{k=1}^K a_k}{\lambda} = \frac{1}{\lambda} \geq \sum_{k=1}^K \pi_k = 1.$$

Hence $\lambda \leq 1$. This is a contradiction to (4.19), so the proof is complete.

(ii) If $m = 0$, we proved $\boldsymbol{\pi} = \boldsymbol{a}$ in Theorem 3.1. Now assume $a_m \leq \epsilon$ and $m > 0$. We can define $\lambda = 1, \mu_k = 0$ and $\boldsymbol{\pi} = \boldsymbol{a}$. It is easy to see that the KKT conditions (4.18) are satisfied. Furthermore, the objective function of (4.17) is strictly convex in its domain and $\boldsymbol{\pi} = \boldsymbol{a}$ is a feasible solution, thus $\boldsymbol{\pi} = \boldsymbol{a}$ is the unique solution. □

By using the same arguments in the proof of Lemma 4.1, we can conclude that $0 < \pi_1 \leq \pi_2 \leq \ldots \leq \pi_K$ when $\boldsymbol{\pi}$ is a solution of (4.17). For a given m, we need to identify a break-point $\hat{k}$ where

$$\pi_k < \epsilon, \text{ if } k \leq \hat{k} \tag{4.20}$$

$$\pi_k = \epsilon, \text{ if } \hat{k} < k \leq m. \tag{4.21}$$

From the KKT conditions (4.18), we have $\pi_k = \frac{a_k}{\lambda}$ for any $k \leq \hat{k}$ and $k > m$. It holds that

$$\begin{aligned} 1 = \sum_{k=1}^K \pi_k &= \sum_{k \leq \hat{k} \text{ or } k > m} \pi_k + \sum_{\hat{k} < k \leq m} \pi_k \\ &= \sum_{k \leq \hat{k} \text{ or } k > m} \frac{a_k}{\lambda} + (m - \hat{k})\epsilon. \end{aligned}$$

Hence, we have $\lambda = \big(\sum_{k \leq \hat{k} \text{ or } k > m} a_k\big) / \big(1 - (m - \hat{k})\epsilon\big)$. For any $k \leq \hat{k}$ or $k > m$, one has

$$\pi_k = \frac{a_k(1 - (m - \hat{k})\epsilon)}{\sum_{i \leq \hat{k} \text{ or } i > m} a_i}, \tag{4.22}$$

which is dependent of $\hat{k}$. For any $\boldsymbol{\pi}$ defined as in (4.21) and (4.22), it is satisfied all of the optimality conditions of (4.17) provided that we can guarantee the feasibility condition $\pi_k \leq \epsilon$ for every $1 \leq k \leq m$ and $\mu_k \geq 0$ for any $\hat{k} < k \leq m$. Note that because $\{a_k\}$ is an increasing sequence and from (4.22) we have $\pi_1 \leq \pi_2 \leq \ldots \leq \pi_{\hat{k}}$. It suffices to find a value $\hat{k}$ such that $\pi_{\hat{k}} < \epsilon$. Let t be the index satisfying

$$a_1 \leq \ldots \leq a_{t-1} \leq a_t < a_{t+1} \leq \ldots \leq a_m$$

where $a_t < \epsilon$, and $a_{t+1} \geq \epsilon$. From Lemma 4.2 we claim that if there exists $\hat{k} \geq 1$ such that $\pi_k < \epsilon$ whenever $k \leq \hat{k}$ then $\hat{k} \leq t$. We should start to search for $\hat{k}$ from t to 1. Checking $\mu_{\hat{k}+1} \geq 0$ is equivalent to verifying

$$\frac{a_{\hat{k}+1}}{\epsilon} \geq \lambda = \frac{\sum_{k \leq \hat{k} \text{ or } k > m} a_k}{1 - (m - \hat{k})\epsilon}.$$

That is, $a_{\hat{k}+1}(1 - (m - \hat{k})\epsilon) \geq \epsilon \sum_{k \leq \hat{k} \text{ or } k > m} a_k$.

Now we are ready to present an algorithmic framework denoted by `SWSA` for solving the problem (3.8) in Algorithm 1 under the assumption (4.12).

By using techniques from convex optimization including exploitation of KKT conditions, a conditional closed-form expression for each mixture component has been derived (fixing the number of components). The point estimate can be determined by combining the closed-form expression for a fixed number of active components t and exhaustive search over t. By the above analysis, we can state the following complexity result:

Algorithm 1 Sparse Weight Selection Algorithm `SWSA`($\boldsymbol{a}$, τ, ϵ)

Set $f_{min} \leftarrow -\sum_{k=1}^K a_k \ln(a_k) + n\tau$
for $m = 0, 1, \ldots, n-1$ **do**
  if $m = 0$ or $a_m \leq \epsilon$ **then**
    $\boldsymbol{\pi} \leftarrow \boldsymbol{a}$
  else
    Find $t \leq m$ such that $a_t < \epsilon \leq a_{t+1}$
    if $t = \emptyset$ **then**
      $\pi_k \leftarrow \begin{cases} \epsilon, & \text{if } k \leq m \\ \frac{a_k(1-m\epsilon)}{\sum_{i=m+1}^n a_i}, & \text{otherwise} \end{cases}$
    else
      for $\hat{k} = t, t-1, \ldots, 1$ **do**
        $\pi_k \leftarrow$ Eqs. (4.21) and (4.22)
        if $\Big(\pi_{\hat{k}} < \epsilon$ and $a_{\hat{k}+1}(1 - (m - \hat{k})\epsilon) \geq \epsilon \sum_{i \leq \hat{k} \text{ or } i > m} a_i\Big)$ **then**
          break
        end if
      end for
    end if
    $g(\boldsymbol{\pi}) \leftarrow -\sum_{k=1}^K a_k \ln(\pi_k) + \tau |\{i : \pi_i > \epsilon\}|$
    if $g(\boldsymbol{\pi}) < f_{min}$ **then**
      $f_{min} \leftarrow g(\boldsymbol{\pi})$ and $\boldsymbol{\pi}^* \leftarrow \boldsymbol{\pi}$
    end if
  end if
end for
return $\boldsymbol{\pi}^*$

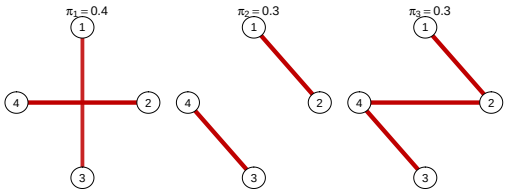

Figure 1: True precision matrices for synthetic data

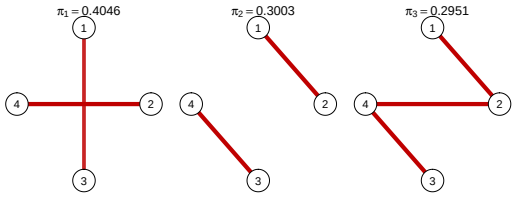

Figure 2: Learned precision matrices by `SWSA`

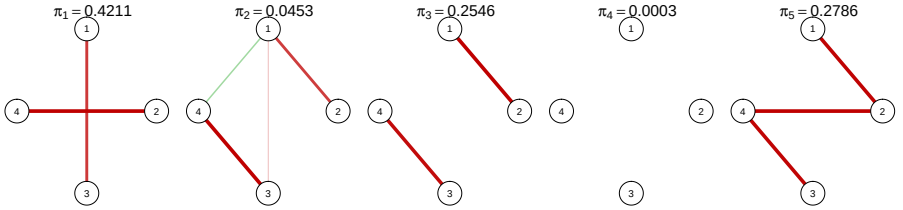

Figure 3: Learned precision matrices by the conventional ARD - `CARD`

THEOREM 4.1. *Algorithm 1 can find a global optimal solution of Eq.* (3.8) *in quadratic time in terms of the maximum number of mixture components* K.

5 Experiments

In this section, we first show the recovery accuracy and convergence behavior of our sparse mixture weight model `SWSA` on synthetic data sets. We then access the performance of our method in terms of the accuracy in anomaly detection and the held-out log likelihood for real data sets.

To be specific and as simple as possible, we assume $p(\boldsymbol{x}|\boldsymbol{\theta}_k)$ to be the Gaussian distribution $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \Lambda_k^{-1})$, where $\boldsymbol{\mu}_k$ and Λ_k are the mean and the precision matrix of the k-th component, respectively. To avoid potential numerical instabilities, we use the Laplace distribution for Λ_k as $P(\boldsymbol{\Theta})$ and use the MAP estimated value for Λ_k for `SWSA` and the conventional automatic relevance determination method (denoted by `CARD`) [5]. The k-means clustering algorithm was used to initialize the variational Bayes methods. The precision matrix Λ_k is initialized by the graphical lasso algorithm [6] for each cluster.

In `SWSA`, we chose $\epsilon = 10^{-4}$. The model parameter τ in (3.8) was selected so that the Bayesian Information Criterion (BIC) score [17] is maximized

$$BIC = 2L(\boldsymbol{x}|\boldsymbol{\pi}, \Theta) - log(N)d,$$

where $L(\boldsymbol{x}|\boldsymbol{\pi}, \Theta)$ is the log-likelihood and $d = KM(M+3)/2 + K + 1$ is the number of free parameters. To check the convergence, we monitor the error at the t-th iteration of the VB iteration defined as

$$\texttt{err} = \|\boldsymbol{\pi}^t - \boldsymbol{\pi}^{t-1}\|$$

until reaching at the termination limit $\texttt{err} < 10^{-6}$.

5.1 Synthetic data To check the utility of `SWSA` in pruning irrelevant components as well as convergence rate, we randomly and independently generated $N = 3\,000$ samples from three distinctive four-variate Gaussian distributions, say, A, B, and C. Their corresponding precision matrices are shown in Fig. 1 and the true mixture weights are given by $\boldsymbol{\pi} = (0.4, 0.3, 0.3)$. The first component has the mean $(5, 0, 0, 5)^\top$, we use the same mean of $(0, 5, 5, 0)^\top$ for the second and third components.

Figures 2 and 3 show the learned precision matrices in terms of the partial correlation coefficients and mixture weights from the proposed method `SWSA` and the conventional ARD `CARD`. Our method precisely recovers the pattern A, B, and C, from left to right, and the mixture weights are also essentially consistent to the true values. The correct number of components $K = 3$ is automatically found as opposed to the provided initial condition of $K = 10$, thanks to the explicit sparsity enforcement. On the other hand, the conventional ARD generates an incorrect number of components $K = 5$, which results in two superfluous components: the second and the fourth ones in Fig 3. We note that the associated mixture weights related to the unwanted components are relatively small (0.0453 and 0.0003), which are removed by our sparsity-promoting technique.

Figure 4 compares the behaviors of `err` and the log-likelihood progresses for the proposed and the conventional ARD approaches. We observe that the conventional approach sometimes gets stuck at local minima, while `SWSA` can stably find the right solution with a much smaller number of iterations, as shown in the figure (368 versus 655 iterations). We notice that the error and log-likelihood curves for `SWSA` are quite bumpy. One explanation is that the sparsity-promoting formula (3.8) effectively enforces some small weights suddenly fall be-

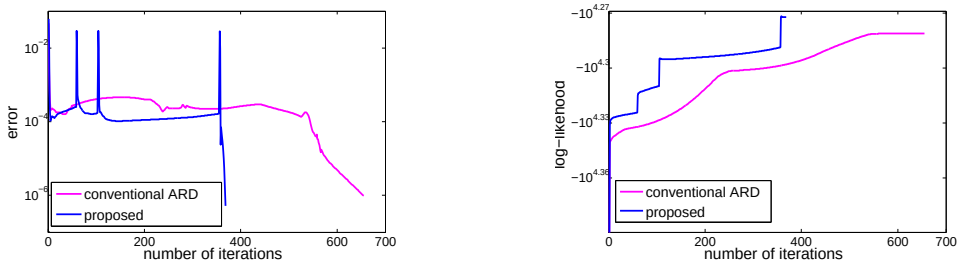

Figure 4: Comparison of convergence rates when the conventional method CARD failed

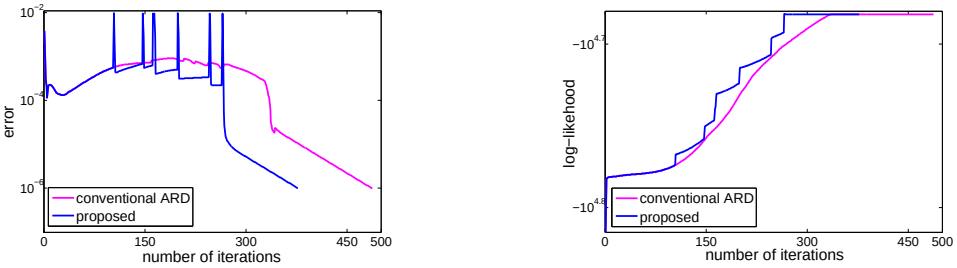

Figure 5: Comparison of convergence rates when both SWSA and CARD were successful

low ϵ at some intermediate iterations, which correspond to big jumps. Note that for this experiment, our method obtained a better log-likelihood (-18 721.5 vs -19 108.1). The smooth curve of the conventional approach suggests that the conventional algorithm strongly encourages convergence by forcing small components to be even smaller. Although Fig. 4 is just for one instance, in our repeated experiments with different random number seeds for the data with this size, CARD produced a noticeably worse solution in most cases.

We also tested a data set, on which both methods converged to the same optimal solution and produced correct estimate. We used a data instance with the same ground truth as before but larger sample size, in particular, consisting of 10000 samples. The result in Fig. 5 shows the same trend as discussed in the previous experiment, the convergence of the proposed method is faster (377 versus 487 iterations).

5.2 Anomaly detection We compare the proposed method SWSA with CARD, principal component analysis (denoted by PCA) [13], and the mean Hotelling's T^2 statistic (denoted by T2) [7] for anomaly detection task. From the learned predictive distribution $p_{pred}(\boldsymbol{x}) = \sum_{k=1}^{K} \pi^k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \Lambda_k^{-1})$, the anomaly score for a new observation is computed by

$$a(\boldsymbol{x}) = -\ln p_{pred}(\boldsymbol{x}).$$

We test on two real-world data sets. The first data set (Wafer) is collected from an etching tool in a superconductor manufacturing process. A semiconductor wafer is processed in an etching chamber and goes through 13 distinctive processing steps. Each of the steps is monitored with 48 sensors such as temperatures, currents, pressures and the flow rate of reactive gases, which can be represented as a vector of length 624(= 13 × 48). Wafer quality (normal/abnomal) is labeled using an on-wafer electric resistance measurement. We used 545 normal samples for training, and 100 samples for testing. For the test set, there are 42 abnormal observations and 58 normal observations. The second data set[1] is a 32-dimensional letter recognition dataset (Letter). We have 100 abnormal observations and 200 normal observations in the test set. The model was trained on 1300 normal data points.

We show the Receiver Operating Characteristics (ROC) curves in Fig. 6 together with the area under the ROC curve (AUC) in Table 1 to analyze the trade-off between the true positive rate and the false positive rate. The model parameters were chosen so that the AUC values are maximized on the test data. We used the number of initial $K = 30$. Upon convergence, SWSA has 18 and 14 active components for Wafer and Letter, respectively; but there are 23 and 17 non-empty clusters for CARD. It can be seen that the ROC curves of SWSA clearly dominate the baselines. Particularly, the AUC values of SWSA outperform that of CARD, but using fewer number of mixture components.

	SWSA	CARD	T2	PCA
Wafer	**0.96**	0.88	0.86	0.81
Letter	**0.97**	0.91	0.85	0.84

Table 1: Comparison of AUC values

[1] http://odds.cs.stonybrook.edu/letter-recognition-dataset/. See [12, 15] for details.

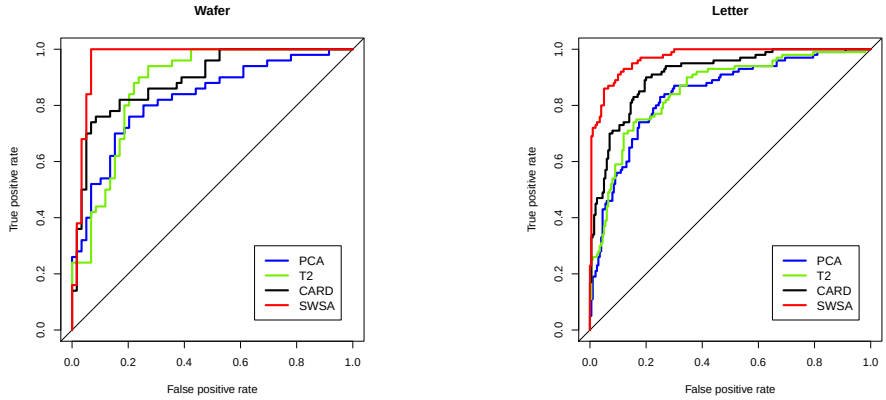

Figure 6: ROC curves

	log-like.	BICscore	Component
`Breast Cancer`			
`SWSA`	**276.75**	1153.72	**5.3**
`CARD`	230.94	907.45	14.2
`CSBDP`	203.51	-	12.6
`VDP`	224.86	-	7.5
`Cloud`			
`SWSA`	**262.88**	2410.21	7.4
`CARD`	228.11	1905.44	10.1
`CSBDP`	249.51	-	7.8
`VDP`	231.02	-	**6.6**
`Parkinsons`			
`SWSA`	-89.02	-3615.72	2.6
`CARD`	-107.53	-5135.02	5.8
`CSBDP`	-98.61	-	6.5
`VDP`	**-86.09**	-	**2.4**
`Anuran Calls`			
`SWSA`	**2592.58**	42528.4	**3.2**
`CARD`	2357.89	37413.6	11.6
`CSBDP`	2426.55	-	13.7
`VDP`	2386.32	-	3.8

Table 2: Performance comparison for held-out log likelihood and the final number of components

5.3 Held-out log likelihood We compare our proposed method `SWSA` with the conventional ARD approach `CARD`, variational Dirichlet process (`VDP`) [3], and collapsed variational stick-breaking Dirichlet process (`CSBDP`) [8] in terms of the log likelihood assigned to held-out data[2]. Hyperparameters of Dirichlet process methods such as initial number of clusters are selected by cross validation. Four real data sets are used to test our algorithm `SWSA`. All of them can be accessed from UCI Machine Learning Repository [9]. We summarize the detailed information as follows.

- The `Breast Cancer` data set from digitized images of a fine needle aspirate of a breast mass has 683 points in 10 dimensional space.
- The `Cloud` data set from cloud cover images is composed of 1024 vectors in 10 dimensions.
- The `Parkinsons` data set consists of 195 individual voice recordings, each record is represented by a 21 dimensional feature vector.
- The `Anuran Calls` data set collected from frog croaking sounds has 7195 points in 22 dimensions belonging to 10 species. We used a major species *HypsiboasCordobae* having 1121 sample points.

In each data set, 90% of samples are used for training and 10% for testing, with which the log likelihood (i.e. $\ln p_{pred}$) is computed as a measure of goodness. The score is averaged over 10 random splits. The VB algorithms start with initial values given by the k-means method for $K = 15$. In Table 2, we show log likelihood for the predictive distributions ("`log-like.`") of the test data, the BIC score ("`BICscore`"), and the final number of components ("`Component`"). A better model should assign higher probability to the test data.

We see that `SWSA` produced final solutions with the number of components much less than that of the conventional ARD approach `CARD`, which illustrates the benefit of adding the ℓ_0 regularization. Our method also gives a higher log likelihood value than that of `CARD`, which demonstrates a better generalization ability achieved by the removal of spurious components. The BIC scores computed on the training data from `SWSA` are also higher than `CARD`.

The Dirichlet process methods `CSBDP` and `VDP` can get very sparse mixture weights, especially `VDP`, but they incur a lower held-out log probability for `Breast Cancer` and `Cloud`. Our method has the best performance in 3 out of 4 instances (`Breast Cancer`, `Cloud`, and `Anuran Calls`), and is ranked second in the remaining (`Parkinsons`). Overall, `SWSA` often gets the sparsest solution (`Breast Cancer` and `Anuran Calls`) and very sparse one for the others.

[2]The codes are available at `https://sites.google.com/site/kenichikurihara/academic-software/variational-dirichlet-process-gaussian-mixture-model`

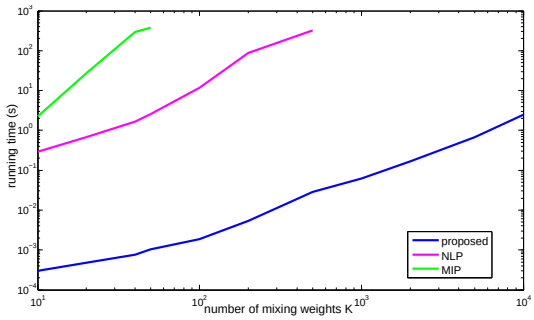

Figure 7: Running times versus the maximum number of components K

5.4 Scalability comparison for `SWSA` We compare the running times between our algorithm `SWSA` for solving the problem (3.8) with the mixed-integer nonlinear programming solver SCIP [1] (denoted by "`MIP`"), and a method using the nonlinear programming solver IPOPT [20] (denoted by "`NLP`") to solve the subprolem (4.16). We randomly generated the vector $\boldsymbol{a}$ as follows: $0.2K$, $0.4K$ and $0.4K$ elements of $\boldsymbol{a}$ are uniformly distributed random numbers between $[1\text{e}{-7}, 1\text{e}{-3}]$, $[1\text{e}{-4}, 1]$, and $[1, 100]$, respectively. We choose $\tau = 0.1$ and $\epsilon = 1\text{e}{-4}$. Our algorithm was written in Matlab and run on a ThinkPad W540 laptop using MATLAB 8.3 with a 64-bit Windows 7 with Intel i7-4800MQ 2.7GHz CPU and 16GB of RAM. The result is summarized in Fig. 7 as a function of the maximum number of components K. We observe that our algorithm scales well with the dimension, and run much faster than other approaches.

6 Conclusions

This paper introduces a novel method for automatic selection of sparse mixture weights in probabilistic mixture models as an alternative to the conventional ARD approach. Based on the new concept ϵ-sparsity, we have proposed a mathematical formula based on ℓ_0 regularization for mixture weight update. We present an efficient algorithm to solve the model in quadratic time. Using synthetic data, we have shown that the proposed method was able to stably identify the correct number of mixture components with a fewer number of iterations, while the conventional ARD got stuck with local minima. For real data sets, our method used a small number of mixture components but often gave better performance in anomaly detection as well as held-out log likehood over existing methods.

References

[1] T. Achterberg. Scip: solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[2] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

[3] D. M. Blei and M. I. Jordan. Variational inference for dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–143, 03 2006.

[4] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[5] A. Corduneanu and C. M. Bishop. Variational bayesian model selection for mixture distributions. In *Artificial intelligence and Statistics*, pages 27–34, 2001.

[6] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[7] T. Idé, D. T. Phan, and J. Kalagnanam. Change detection using directional statistics. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 1613–1619, 2016.

[8] K. Kurihara, M. Welling, and Y. W. Teh. Collapsed variational dirichlet process mixture models. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2796–2801, 2007.

[9] M. Lichman. UCI machine learning repository, 2018.

[10] G. Malsiner-Walli, S. Frühwirth-Schnatter, and B. Grün. Model-based clustering based on sparse finite gaussian mixtures. *Statistics and Computing*, 26(1):303–324, 2016.

[11] G. Malsiner-Walli, S. Frühwirth-Schnatter, and B. Grün. Identifying mixtures of mixtures using bayesian estimation. *Journal of Computational and Graphical Statistics*, 26(2):285–295, 2017.

[12] B. Micenková, B. Mcwilliams, and I. Assent. Learning outlier ensembles: The best of both worlds supervised and unsupervised. In *KDD ODD2 Workshop*, 2014.

[13] S. Papadimitriou and P. Yu. Optimal multi-scale patterns in time series streams. In *Proc. ACM SIGMOD Intl. Conf. Management of Data*, pages 647–658, 2006.

[14] C. E. Rasmussen. The infinite gaussian mixture model. In *Advances in neural information processing systems*, pages 554–560, 2000.

[15] S. Rayana and L. Akoglu. Less is more: Building selective anomaly ensembles. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(4):42:1–42:33, May 2016.

[16] A. Roychowdhury and B. Kulis. Gamma processes, stick-breaking, and variational inference. In *AISTATS*, 2015.

[17] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[18] J. Sethuraman. A constructive definition of dirichlet priors. *Statistica sinica*, pages 639–650, 1994.

[19] S. A. Vavasis. *Nonlinear Optimization: Complexity Issues*. Oxford University Press, Inc., New York, NY, USA, 1991.

[20] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.