Stokes, Gauss, and Bayes walk into a bar...

by

E. P. Kightley

B.S., University of Cincinnati, 2013

M.S., University of Colorado Boulder, 2016

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Applied Mathematics

2019

This thesis entitled:
Stokes, Gauss, and Bayes walk into a bar...
written by E. P. Kightley
has been approved for the Department of Applied Mathematics

_____

Prof. Stephen Becker

_____

Prof. Jem Corcoran

_____

Prof. Manuel Lladser

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Kightley, E. P. (Ph.D., Applied Mathematics)

Stokes, Gauss, and Bayes walk into a bar...

Thesis directed by Prof. Stephen Becker

This thesis consists of three distinct projects. The first is a study of microbial aggregate fragmentation, in which we develop a dynamical model of aggregate deformation and breakage and use it to obtain a post-fragmentation density function. The second and third projects deal with dimensionality reduction in machine learning problems. In the second project, we derive a one-pass sparsified Gaussian mixture model to perform clustering analysis on high-dimensional streaming data. The model estimates parameters in dense space while storing and performing computations in a compressed space. In the final project, we build an expert system classifier with a Bayesian network for use on high-volume streaming data. Our approach is specialized to reduce the number of observations while obtaining sufficient labeled training data in a regime of extreme class-imbalance and expensive oracle queries.

Keywords: microbial flocculation; dimensionality reduction; random projections; Gaussian mixture models; clustering; Bayesian networks; probabilistic graphical models; anomaly detection; classification

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

This thesis is a compilation of three distinct projects. In the first project, "Fragmentation Distribution for Microbial Aggregates in Shear Flow" [Chapter 2], we develop a modeling framework to simulate microbial aggregate fragmentation, and then apply the model to compute a post-fragmentation density function. In the second project, "One-Pass Sparsified Gaussian Mixtures" [Chapter 3], we derive a clustering algorithm for use in conjunction with a dimensionality reduction scheme. In the third project, "Oracle Epiphany in Bayesian Networks" [Chapter 4], we present a method to simultaneously obtain labels for use in supervised learning and reduce the number of datapoints for a classification task using Bayesian networks.

Chapter 2 is a stand-alone project independent of the rest of the thesis. Chapters 3 and 4 are related in that both have to do with applying machine learning methods in a regime characterized by high volumes of data. In both cases, the fundamental difficulty we face is that the data are too large to process. Let $\{\mathbf{x}_i\}_{i=1}^N$ be a set of observations, with $\mathbf{x}_i \in \mathbb{R}^P$. Either the number of data points (N) or the latent dimension (P), or both, could be prohibitively large. In chapter 3 we develop a technique to perform clustering analysis while reducing the latent dimension of the data: we reduce the size of each datapoint $\mathbf{x}_i \in \mathbb{R}^P$ from P to Q with $Q \ll P$. We do so in the context of streaming data and our method is **one-pass**, meaning that we obtain estimates for statistics in the full P-dimensional space while storing and performing computations in the much-smaller Q-dimensional space. In chapter 4 we concern ourselves with the orthogonal problem of reducing the number of datapoints N. We do so in the domain-specific context of classifying cyber security

threats, where we face the additional problems of a severe paucity of labeled data, a high cost to obtaining a label for a given observation, and extreme class imbalance. To address these we combine recent work on **oracle epiphany** with **Bayesian networks** to reduce the number of datapoints we need to consider while simultaneously increasing the relative frequency of the minority class and facilitating the expensive process of label acquisition.

Here we briefly introduce each project, providing motivation and a summary of our main findings and contributions. Each project is then discussed independently in the subsequent chapters.

1.1 Project 1: Fragmentation Distribution for Microbial Aggregates in Shear Flow

1.1.1 Motivation

Microbial flocculation refers to the process by which single-celled microorganisms intermittently persist in multicellular aggregates suspended in an aqueous solution, a phenomenon ubiquitous in industry and nature. In industrial applications the goal is either for the microbes to consume an unwanted substance, such as in wastewater treatment, or to excrete a desired substance, such as in beer fermentation and algal biofuel production. Typically, the microbes do some portion of their work as free-floating individual cells, but begin to clump together as the process proceeds closer to completion. In beer brewing, for instance, yeast cells aggregate as the alcohol content grows, and ultimately settle out of solution.

The dynamics of how these aggregates form and break apart, including their distribution in time, space, and size, can have a significant impact on the effectiveness of the intended process. If the aggregates grow too quickly, they can settle out of solution before they have converted enough of the contents of the solution. There are many mechanical and chemical procedures to prevent this; of particular interest to us is the mechanical mechanism of shear flow. On the other hand, care must be taken to ensure that these safeguards do not prevent aggregation entirely, since it is usually necessary to remove the microbes from the suspension before the product can be further

processed.

Significant efforts have therefore been devoted to the study of flocculation dynamics. The component of these dynamics in which we are interested here is aggregate breakage or **fragmentation**. Specifically we want to study whether aggregates tend to break into equally-sized daughters (**splitting**) on one extreme or one large and one small daughter (**erosion**) on the other. To simulate fragmentation we develop a model of aggregate fragmentation in shear flow and to use this model to estimate a **post-fragmentation density function**, a distribution for aggregate size as shear flow exhaustively breaks apart an aggregate into smaller and smaller daughters.

1.1.2 Main Contributions

We develop a dynamical model of aggregate dynamics in shear flow. To do so we couple constituent models for the deformation of a droplet in shear flow and for the surface forces on a solid ellipsoid. Our model tracks the position, velocity, shape, deformation, and surface force density on an idealized aggregate tumbling in shear flow. We use this model to simulate aggregate fragmentation by choosing to break the aggregate at predetermined locations if the surface force exceeds a minimum threshold. We test three different fragmentation regimes, finding that erosion tends to be the dominant fragmentation mechanism. This work is published in [1].

1.2 Project 2: One-Pass Sparsified Gaussian Mixtures

1.2.1 Motivation

Clustering is a common statistical data analysis task in which observations are grouped together according to some indicator of similarity. In the context of machine learning we often seek to assign observations to clusters as well as to learn some parametric representation of these clusters. We may wish to use this process as a generative model from which to sample more data, to classify or cluster future observations, as a feature engineering tool, or as a component of exploratory analysis and descriptive statistics.

It is often the case that we will wish to reduce the size of the dataset, both to reduce storage space as well as to speed up data processing. There is a vast and growing range of methods to do so. Our approach is inspired by the classical Johnson-Lindenstrauss lemma and its applications in the field of compressed sensing. Modern interpretations of the lemma guarantee that certain low-dimensional embeddings $\Omega : \mathbb{R}^P \to \mathbb{R}^Q$, with $Q < P$, will preserve relative pairwise distances between points to within a small tolerance with high probability. These embeddings can be computed and applied quickly, typically in $\mathcal{O}(P \log P)$ operations per datapoint as FFT-like transformations. The idea is thus to pay an upfront cost of $\mathcal{O}(NP \log P)$ so that in subsequent computations we can replace $P \mapsto Q$. These subsequent computations can often be quadratic or cubic in P (e.g. $P \times P$ inversions), and hence the overall algorithm speed can be improved significantly even including the initial $\mathcal{O}(P \log P)$ cost.

In the classical regime, one potential drawback of such compression is that it is usually impossible to recover statistics in the original P-dimensional space without accessing all of the original, dense data. This requires storing that data, as well as whatever computational cost is needed to obtain the statistic. Such is the case for statistics as simple as the sample mean. Our line of work avoids this problem by using a different projection $P \mapsto Q$ (which we call **sparsification**) for each datapoint $\mathbf{x}_i$, so that we can perform approximate computations in the dense P-dimensional domain using only Q-dimensional objects. We therefore do not need to access the original data ever again after obtaining the projections, whence the phrase **one-pass**. The material presented in Chapter 3 builds on previous work from our group in which the one-pass projection scheme was developed and various probabilistic bounds were established on the low-dimensional embeddings. In that work the scheme was applied to k-means clustering, a simple and yet often very effective clustering algorithm. Our work here extends this concept to a more sophisticated clustering model, Gaussian mixture models.

1.2.2 Main Contributions

We present an algorithm to perform sparsified one-pass Gaussian mixture modeling (SGMM). This algorithm uses Q-dimensional embeddings of P-dimensional data (N datapoints) to fit a K-component Gaussian mixture model in $\mathcal{O}(KNQ)$ operations for diagonal or spherical covariances. The algorithm uses Expectation-Maximization to find a local stationary point of the likelihood function. We derive the maximum likelihood estimators for model parameters under sparsification. On example data, SGMM is able to recover over 90% peak accuracy for a GMM using $< 10\%$ of the data ($Q/P < 0.1$). This work has been submitted to an IEEE conference and the manuscript is available on arXiv [2].

1.3 Project 3: Oracle Epiphany in Bayesian Networks

1.3.1 Motivation

Perhaps the most common goal in machine learning is the task of classification: given an observation x and possibly many classes $\{T_1, T_1, \ldots, T_k\}$, we want to know of which class x is an example. This is conceptually related to the clustering task discussed in the preceding chapter; however, classification is typically a supervised task, in the sense that we assume k meaningful and distinct classes, and we have examples for each from which we will learn a predictive model, called a classifier, that we can use to predict the class of unseen examples.

Classification problems arise in many fields and applications, such as computer vision and natural language processing. Typically a classifier is a parametric model with parameters that are inferred from a training set of labeled examples. It is often the case, however, that labeled training examples are very rare, even when unlabeled examples are ubiquitous. Several approaches have been developed to address this problem, including one-shot learning in computer vision, generative adversarial networks, and oracle query methods in active learning.

In this chapter we consider developing a classifier in a regime starting with no labeled data. Our goal is domain-specific: we want to build a classifier to detect cyber security threats. In

addition to the problem of having no labeled data, we also face insurmountably high volumes of streaming data and extreme class imbalance. Furthermore, the process to obtaining a label for a given observation is extremely expensive. To address these issues we build a Bayesian network classifier and use it in an iterative fashion, in an approach similar to feature engineering, in order to reduce the number of datapoints while increasing the relative frequency of the minority class in the training set.

1.3.2 Main Contributions

We build a Bayesian network using domain expert knowledge for use as a classifier in a regime characterized by extreme class imbalance and expensive oracle queries. We do so using an iterative approach to model construction and label acquisition, leveraging the concept of oracle epiphanies to simultaneously increase the relative frequency of the minority class and decrease the number of observations, each by two orders of magnitude. We apply the process to obtain a labeled dataset, train the Bayesian network classifier, and obtain high accuracy on the training and test sets (7/8 minority events correctly identified) while compressing the dataset from $>$ 200K to 494 observations.

Chapter 2

Fragmentation Distribution for Microbial Aggregates in Shear Flow

This chapter is adapted from [1]. Research was supported by the following National Science Foundation grants and fellowships: NSF IGERT 1144807, NSF GRFP DGE 1144083, and NSF DMS 1225878.

We present a model for the force acting to fragment a biofilm-seeded microbial aggregate in shear flow, which we derive by coupling an existing model for the shape and orientation of a deforming ellipsoid with one for the surface force density on a solid ellipsoid. The model can be used to simulate the motion, shape, surface force density, and breakage of colloidal aggregates in shear flow. We apply the model to the case of exhaustive fragmentation of microbial aggregates in order to compute a post-fragmentation density function, indicating the likelihood of a fragmenting aggregate yielding daughter aggregates of a certain size.

2.1 Introduction

Microbial flocculation, the process whereby single-celled microbes in suspension persist as multicellular aggregates for a portion of their life-cycle, is ubiquitous in nature and industry [3, 4, 5, 6]. Important applications of flocculation include beer fermentation [7, 8, 9, 10], wastewater treatment [11, 12, 13], and biofuel production [14, 15, 16, 17]. In the fermentation of beer, yeast cells consume sugar in wort and excrete alcohol (among other things not nearly as interesting to a stressed PhD student). During this process the free-floating yeast cells will eventually flocculate into large aggregates and sink to the bottom of the liquid. The point at which they do so is critical

to the quality of the beer produced: if the aggregates form too early, not enough of the sugars will have been consumed, and the beer will be sweet, or even worse, not very strong. Conversely, if the aggregates do not flocculate sufficiently, the flavor and quality will suffer (rest easy here though; it seems the beer still works in this case). The rate of flocculation and sedimentation are traits selected for through (un)natural selection, and this topic continues to be an active area of research [9, 8].

A critical but often under-appreciated component in the life-cycle of beer consumption is wastewater treatment, which also depends on microbial flocculation [11, 12, 13]. Bacteria are introduced into wastewater in order to consume sludge[1] and remove phosphorus and other nutrients. A variety of species of both aerobic and anaerobic bacteria are used, and the ecosystem must be delicately managed to ensure proper treatment; in particular, the size and shape of these aggregates, as well as the rate at which they form, are important factors [18, 19, 20, 21].

The same concepts apply to algal biofuel production, where algae is used to convert solar energy into fuel [14, 15, 16, 17]. The algae cells flocculate throughout the process, and it is again important to maintain appropriate size distributions of aggregates for optimal efficiency. Considerable attention has therefore been devoted to modeling the dynamics of flocculating [22, 23, 24, 25]. One popular approach is to solve some variation of a population-balance equation (PBE) for the size distribution of aggregates as a function of space and time [26].

Such models generally account for the fact that the microbial aggregates can break apart, for example with the inclusion of a fragmentation kernel and post-fragmentation distribution in the population balance equation. Microbial aggregate fragmentation, however, remains a relatively poorly understood phenomenon. There exists a rich literature in rheology devoted to the study of fluid-fluid emulsions, and in particular to modeling the breakage and resulting size distributions of dispersed droplets (see [27] for a review), and for this reason microbial aggregates are sometimes treated like ellipsoidal (hydrodynamically equivalent) droplets for the purposes of approximating their shape and motion [28, 29, 30, 31, 32]. It is not clear, though, that the corresponding breakage

[1] This is, surprisingly, actually the technical term.

and size-distribution models from the rheology literature on emulsions are equally applicable. The inhomogenous nature of microbial aggregates may mean that some breakage patterns are more likely than others, and we may wish to use knowledge about the structure and composition of the colloid when modeling how and where they break.

This current work is a part of our long term efforts to develop a more accurate fragmentation model for use in PBE-based models. We have approached this problem using both **bottom-up** microscale modeling of individual flocs (this work and [33]) as well as a complementary **top-down** inverse problem methodology. The top-down approach takes time series of aggregate size-distributions and infers a post-fragmentation distribution [34] but does not incorporate any information about the heterogeneities in individual flocs. Here we extend the bottom-up approach that we initially proposed in [33]. As described in [33], we have used confocal microscopy to identify 3D positions of bacteria in a small number (39) of suspended aggregates. However, it is infeasible to experimentally obtain these types of 3D images for large populations of flocs. In [35] we used the 3D positions of bacteria in an aggregate to simulate the tumbling and deformation of bacterial flocs in laminar flow. However, it is also infeasible to perform large numbers of these simulations. Accordingly, we have pursued a hybrid approach merging models for the physics of viscous ellipsoids in flow with an analysis of the locations of high negative Gaussian curvature in the polysaccharides encapsulating the microbes.

Toward this end, we present a model to compute the force acting to break a microbial aggregate at a specified location. We call **aggregate fragmentation** the process in which a parent aggregate containing m microbes (its **size**) breaks into two **daughters** of sizes k and $m-k$. This is an extension of our earlier work in which we began to develop a framework for identifying likely breakage locations [33], and the present work expands upon this by introducing deformation to the model and by refining the computation of the fragmentation force. We construct our model by coupling a model for the deformation of a fluid droplet [36, 37] (hereafter, the **Deformation Constituent Model** or DCM) with one that computes the surface force density on a solid ellipsoid [38] (hereafter, the **Force Constituent Model** or FCM). We restrict ourselves to the case of

viscous shear under the assumption of Stokes' flow, and our choice of deformation model is further guided by the requirements that (1) the surface remain ellipsoidal and (2) there be a restorative force (in this case interfacial tension) acting to oppose the deformation imposed by the shear field.

We then apply the model to the problem of generating a **post-fragmentation density function** $\Gamma(k \mid m)$, a conditional distribution giving the probability that an aggregate fragmentation event will yield a daughter aggregate of size k given that a parent of size m fragments. This density is often assumed to be normal or log-normal in the literature. The normal distribution is a model for the single event of an aggregate fragmenting into two (roughly) evenly-sized daughters. The log-normal distribution is a model for the cumulative effect of repeated fragmentation events (all with a normal post-fragmentation distribution) exhaustively until no more events are possible. We refer to these forms of fragmentation as **splitting**. In our previous work we obtained a form of $\Gamma(k \mid m)$ corresponding to fragmentations yielding one daughter much larger than the other, which we refer to as **erosion** [33]. Example distributions following each of these fragmentation mechanisms can be seen in Figure 2.1. In our previous simulations we found that larger aggregates tended to fragment by erosion whereas smaller ones tended to fragment by splitting. Here we reproduce and expand upon these simulations with the improved fragmentation force model.

The paper is organized as follows. In Section 2.2 we present the methods: first describing the DCM and FCM constituent models and how we couple them (Section 2.2.1), then using the coupled models to define the concept of fragmentation force (Section 2.2.2), and lastly explaining how we apply this concept to microbial aggregate fragmentation (Section 2.2.3). In Section 2.3 we present the results of our simulations, beginning with an exploration of the motion and orientation of an aggregate under the DCM (Section 2.3.1), followed by a discussion of the fragmentation force (Section 2.3.2), and the application of this model to microbial aggregate fragmentation and the generation of post-fragmentation density functions (Section 2.3.3). We finish with some concluding remarks as well as a brief outline of our plans for upscaling these results for use in a PBE equation (Section 2.4).

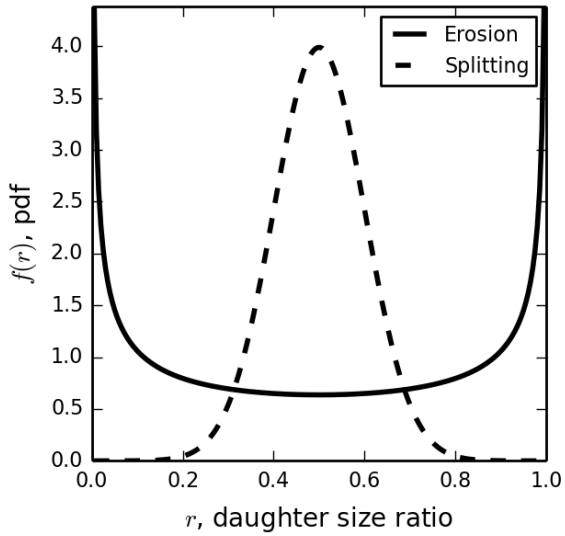


Figure 2.1: Example post-fragmentation density functions corresponding to erosion and splitting fragmentation mechanisms.

2.2 Methods

2.2.1 Constituent Models

Deformation Constituent Model (DCM)

The DCM is the model we use to describe the deformation and rotation motion of a fluid ellipsoid in simple shear flow. This model is developed in [37, 36], to which the reader is referred for a full development. Here we present only the material necessary for the coupling of the models. An arbitrary ellipsoid centered at the origin can be represented by a shape tensor, a symmetric 3×3 tensor $\mathbf{G}$ such that $\mathbf{x}^T\mathbf{G}\mathbf{x} = 1$ for any point $\mathbf{x}$ on the surface of the ellipsoid. The shape tensor is orthogonally diagonalizable, so that

$$\mathbf{D} = \mathbf{R}^T\mathbf{G}\mathbf{R} \tag{2.1}$$

where $\mathbf{D}$ is diagonal and $\mathbf{R}$ is a rotation. We can choose to construct $\mathbf{G}$ such that the diagonal entries of $\mathbf{D}$ (the eigenvalues of $\mathbf{G}$) λ_i are defined by $\lambda_i = 1/a_i^2$ where $\mathbf{a} = (a_1, a_2, a_3)$ are the axes lengths of the ellipsoid.

The deformation constituent model consists of an ODE we can solve for such a shape tensor $\mathbf{G}(t)$. Assuming constant volume and Stokes flow in an incompressible Newtonian fluid, the governing equation for shape of the ellipsoid is

$$\dot{\mathbf{G}} + \mathbf{L}_d^{\mathrm{T}} \cdot \mathbf{G} + \mathbf{G} \cdot \mathbf{L}_d = 0 \tag{2.2}$$

where $\mathbf{G}$ is the shape tensor of the ellipsoid, as described above, $\dot{\mathbf{G}}$ is the material derivative of $\mathbf{G}$, and $\mathbf{L}_d$ is the droplet velocity gradient.

To solve equation (2.2) for $\mathbf{G}$, an expression for $\mathbf{L}_d$ is required; this expression must depend only on the external velocity gradient $\mathbf{L}$, the shape and orientation of the ellipsoid, and the input parameters. The reader is referred to [37, 36] for derivations and the precise form of $\mathbf{L}_d$. For our purposes, it suffices to note that the output of solving the DCM will be a time-series of shape tensors $\mathbf{G}(t_i)$ satisfying equation (2.2).

Force Constituent Model (FCM)

Here we describe the FCM developed in [38]. Given an ellipsoid with axes lengths a_i such that $a_1 \geq a_2 \geq a_3$, under the assumption of Stokes' flow, the force density on the surface of a solid ellipsoid in simple shear can be written as

$$\mathbf{f} = \left(-p_0 \mathbf{I} - 4\mu \sum_{j=0}^{3} \chi_j A_j^j \mathbf{I} + \frac{8\mu}{a_1 a_2 a_3} \mathbf{A}^T \right) \mathbf{n} \tag{2.3}$$

where p_0 is the external pressure, μ is the matrix viscosity, a_i are the axes lengths, and $\mathbf{n}$ is normal to the surface of the ellipsoid. The matrix $\mathbf{A}$ is in turn defined by

$$A_k^i = \begin{cases} \frac{3\chi_i'' E_i^i - \sum_{l=1}^{3} \chi_l'' E_l^l}{6(\chi_1''\chi_2'' + \chi_1''\chi_3'' + \chi_2''\chi_3'')} & \text{for } i = k, \\ \frac{\chi_i E_k^i + a_k^2 \sum_{l=1}^{3} \varepsilon^{ikl} \chi_l' (\varepsilon^{ikl} \Omega_k^i + \omega_l)}{2(a_k^2 \chi_k + a_i^2 \chi_i) \sum_{l=1}^{3} |\varepsilon^{ikl}| \chi_l'} & \text{for } i \neq k \end{cases} \tag{2.4}$$

where $\mathbf{E} = \frac{1}{2}(\mathbf{L} + \mathbf{L}^{\mathrm{T}})$ is the rate-of-strain tensor, $\mathbf{\Omega} = \frac{1}{2}(\mathbf{L} - \mathbf{L}^{\mathrm{T}})$ is the vorticity tensor, and ω_l is the lth component of the angular velocity $\boldsymbol{\omega}$ of the ellipsoid. The elliptic integrals χ_j used are defined by

$$\chi_j = \int_0^{\infty} \frac{d\xi}{(a_j^2 + \xi)\sqrt{(a_1^2 + \xi)(a_2^2 + \xi)(a_3^2 + \xi)}} \tag{2.5}$$

with

$$\chi_i^{'} = \frac{\sum_{k,l=1}^{3} \varepsilon^{ikl}(\chi_l - \chi_k)}{\sum_{k,l=1}^{3} \varepsilon^{ikl}(a_k^2 - a_l^2)} \tag{2.6}$$

and

$$\chi_i^{''} = \frac{\sum_{k,l=1}^{3a} \varepsilon^{ikl}(a_k^2\chi_k - a_l^2\chi_l)}{\sum_{k,l=1}^{3} \varepsilon^{ikl}(a_k^2 - a_l^2)}. \tag{2.7}$$

Coupling the Constituent Models

The matrix $\mathbf{A}$ in Equation (2.4) depends upon the matrix velocity gradient $\mathbf{L}$ and the angular velocity $\boldsymbol{\omega}$ of the ellipsoid, both of which must be expressed in a frame of reference relative to the center of the ellipsoid; i.e., one that rotates with respect to the external frame of reference. In the case of a solid ellipsoid in shear flow, there are analytic representations for both of these quantities [38], but in our model, the motion of the ellipsoid is dictated by the deformation constituent model, and we must therefore compute $\mathbf{L}$ and $\boldsymbol{\omega}$ numerically as they do not have closed-form solutions. The rotation connecting the two reference frames is represented by the matrix $\mathbf{R}(t)$ in equation (2.1) which we obtain by diagonalizing the solution $\mathbf{G}(t)$ to the DCM, equation (2.2). In simple shear flow, the velocity gradient $\mathbf{L}$ is constant in time in an external frame of reference. If the ellipsoid rotates according to $\mathbf{R}(t)$ in the external frame, then the shear field rotates according to $\mathbf{R}^T(t)$ in the ellipsoid frame. Thus we set $\mathbf{L}_R(t) = \mathbf{R}(t)\mathbf{L}\mathbf{R}^T(t)$, and use $\mathbf{L}_R$ in equation (2.3). Expressing the angular velocity in the anti-symmetric matrix

$$[\boldsymbol{\omega}(t)]_\times \equiv \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix} \tag{2.8}$$

a straightforward calculation tracking the motion of an arbitrary point on the ellipsoid surface yields the relation

$$[\boldsymbol{\omega}(t)]_\times = \left(\mathbf{R}(t)\mathbf{R}'(t)\right)^T. \tag{2.9}$$

We approximate $\mathbf{R}'(t)$ using the discretized solution $\mathbf{R}(t_i)$ to equation (2.2) and then use equation (2.9) to compute $[\boldsymbol{\omega}(t)]_\times$, giving $\boldsymbol{\omega}$, the angular velocity of the ellipsoid in the external frame. In

the ellipsoid frame, the shear field is rotating in the opposite direction, with angular velocity $-\boldsymbol{\omega}$. This is the quantity we use in equation (2.3).

2.2.2 Fragmentation Force

In this section we develop the concept of the fragmentation force, an approximation to the force acting to pull apart a deformable ellipsoidal particle in shear flow. To do so we will use the coupled models described in the preceding section to compute an integral of the force density on the surface of the ellipsoid. We define the fragmentation force in such a manner as to permit us to specify where we think breakage will occur by way of a **fragmentation plane**, $\mathcal{P}$, which intersects the ellipsoid. We introduced a similar concept in our previous work [33] for use on non-deformable ellipsoidal approximations to microbial aggregates. In practice, the location and orientation of $\mathcal{P}$ is to be chosen based upon structural information about the aggregate. In our aforementioned application, for example, we preferentially chose planes corresponding to locations where we expected the surface of the microbial aggregate to exhibit a more negative Gaussian curvature.

Suppose that we have chosen some plane $\mathcal{P}$ defined by a normal vector $\mathbf{n}_p$ and interior point $\mathbf{x}_p$, so that $\mathbf{n}_p \cdot (\mathbf{x}_p - \mathbf{x}) = 0$. Let $\mathbf{f}(\mathbf{x})$ be the force density at point $\mathbf{x}$ on the surface $\mathcal{E}(t)$ to an ellipsoid at time t, computed from equation (2.3) as described the preceding section. Define the fragmentation force as defined as

$$F = \int_{\mathcal{E}(t)} s(\mathbf{x}, P)\, |\mathbf{f}(\mathbf{x}) \cdot \mathbf{n}_p|\, d\mathbf{x} \tag{2.10}$$

where

$$s(\mathbf{x}, P) = \begin{cases} 1 & \text{if } \mathbf{f}(\mathbf{x}) \text{ acts to pull against } P \\ -1 & \text{if } \mathbf{f}(\mathbf{x}) \text{ acts to push into } P \end{cases}. \tag{2.11}$$

The integrand is thus the signed magnitude of the component of $\mathbf{f}$ acting against $\mathcal{P}$, where s indicates whether this component acts to pull against or push into the plane. Thus s explicitly accounts for the fact that some of the surface force density may in fact compress against the plane

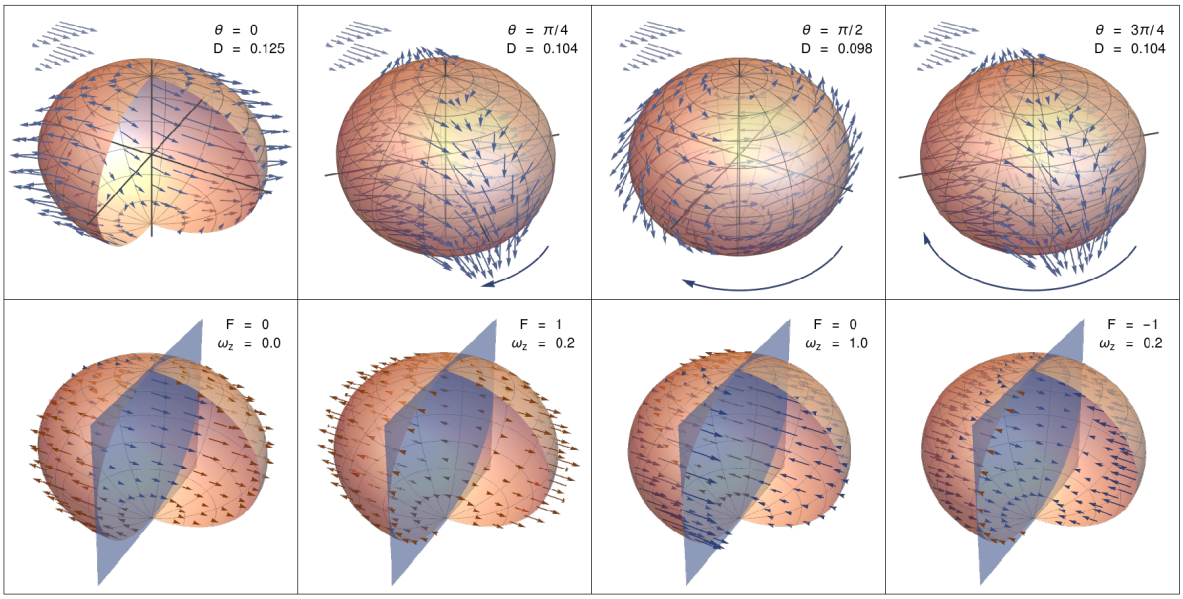


Figure 2.2: A sample ellipsoid shown at four characteristic time points. The ellipsoid is undergoing periodic tumbling with mild deformation. First row: view from an external reference frame, with surface forces and flow field. $D = (a_1 - a_3)/(a_1 + a_3)$ is the Taylor deformation parameter, θ is the angle through which the ellipsoid has rotated. Second row: view from the ellipsoid reference frame, with components of the surface force acting to push into (blue) and pull against (red) a sample fragmentation plane. F is the relative fragmentation force with respect to the plane, and ω_z is the relative angular velocity.

and thus oppose breakage. Figure 2.2 shows snapshots of an ellipsoid in flow along with sample force density vectors, as well as with a sample fragmentation plane and components of the surface force density acting normal to it. As can be seen in the second row of the figure, sometimes the force density acts to compress against the plane (blue arrows) and sometimes to pull against it (red arrows). Figure 2.3 shows the same evolution projected into two dimensions along the $x = 0$ plane.

2.2.3 Aggregate Fragmentation

Having defined the fragmentation force, we now wish to apply it to the case of aggregate fragmentation. To do so we begin with a dataset indicating the coordinates of bacterial centers of mass in 39 **Klebsiella pneumoniae** aggregates (dataset described in [33]). Depicted in Figures 2.4 and 2.5 are two visualizations of a 3D reconstruction of one of the bacterial aggregates. In Figure 2.4, the green surface is the **Klebsiella pneumoniae** cell wall and the blue surface is an estimated

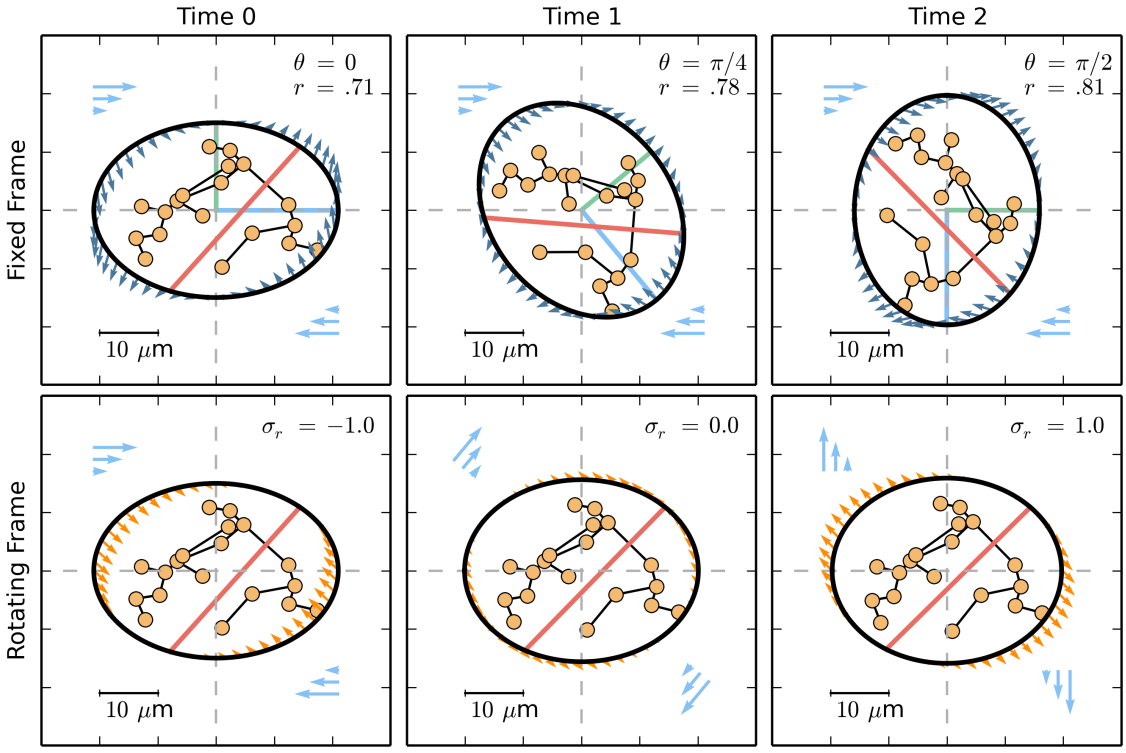


Figure 2.3: A sample ellipsoid shown at four characteristic time points. The ellipsoid is undergoing periodic tumbling with mild deformation. First row: view from an external reference frame, with surface forces and flow field. $D = (a_1 - a_3)/(a_1 + a_3)$ is the Taylor deformation parameter, θ is the angle through which the ellipsoid has rotated. Second row: view from the ellipsoid reference frame, with components of the surface force acting to push into (blue) and pull against (red) a sample fragmentation plane. F is the relative fragmentation force with respect to the plane, and ω_z is the relative angular velocity.

location of the edge of the extracellular polymeric substance (EPS) surrounding the bacteria (1 micron away from the cell wall). Practically speaking, the edge of the EPS is not well-defined and as described in [39], the density of the EPS is highest near the wall and decays within a few microns of the wall. In Figure 2.5, the same floc is shown with the Gaussian curvature (K) computed on the approximate EPS surface. The regions with large negative K are energetically unfavorable for soft matter [40] and are thus the best candidates for potential breakage.

To compute the forces on a tumbling and deforming aggregate is nontrivial.[2] We choose to approximate the forces on the aggregate surface with the forces on a hydrodynamically equivalent

[2] See [35, 41] for illustrations of the challenges of simulating the biomechanics of communities of bacteria.

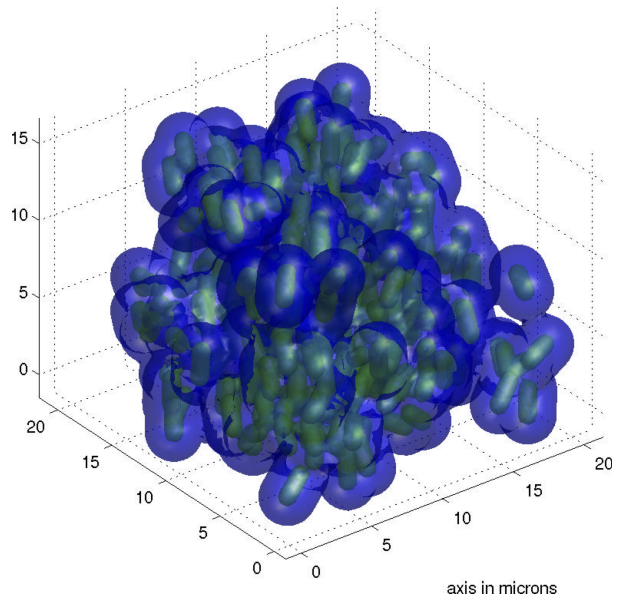


Figure 2.4: Bacterial aggregate reconstructed from confocal microscopy slices. Green surface is the cell wall. Blue surface is the approximate edge of the extracellular polymeric substance (EPS) surface.

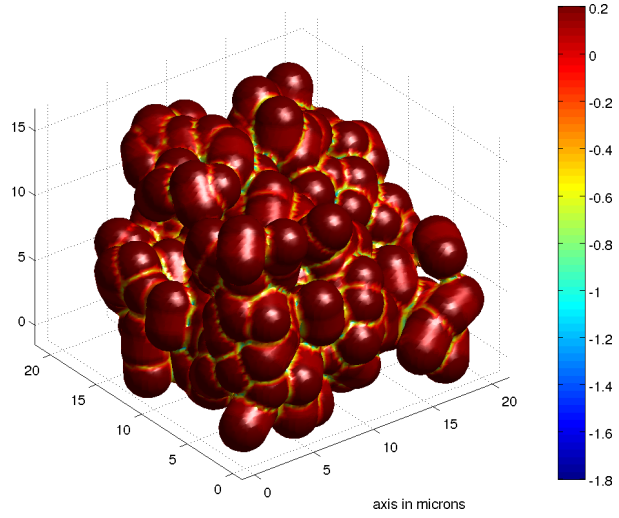


Figure 2.5: Same bacterial aggregate as in Figure 2.4. Colors correspond to the Gaussian curvature computed on the EPS surface. The regions with the most negative curvature are energetically unfavorable and are the best candidates for separation.

ellipsoid. Thus the first task is to approximate an aggregate's surface with an ellipsoid which can then rotate and deform over time according to the deformation constituent model, equation (2.2). At any particular time point, we further wish to specify a fragmentation plane $\mathcal{P}$ based on some structural property of the aggregate, and then use equation (2.10) to compute the force acting to break the aggregate along $\mathcal{P}$. If the force is large enough, the aggregate will fragment into two daughters. We proceed to describe the details of this process. Figure 2.6 shows the various constructs derived below.

Ellipsoidal Representation In order to apply the model we have developed above, we must represent the surface of an aggregate as a hydrodynamically equivalent ellipsoid. The justification for this process is detailed in [33]. Beginning with the coordinates of an aggregate's bacterial centers of mass, we first translate them so they are centered about the origin. We then rotate the coordinates using a principal components decomposition so that the maximal variation lies along the x axis, then the y axis, and finally the z axis of the coordinate system. We take an ellipsoid with axes lengths equal to twice the standard deviation along each axis. The ellipsoid axes lengths specify the initial shape tensor $\mathbf{G}_0$ for the governing equation (2.2), and given the remaining parameters we can then solve this equation to obtain the shape and orientation of the ellipsoid over time, which is fully specified by the axes lengths $\mathbf{a}(t)$ and a rotation $\mathbf{R}(t)$.

Location of Fragmentation Plane In order to check for fragmentation, we must specify the intersecting plane used to compute the force in equation (2.10). To rapidly identify candidate breakage locations, we will create a spanning tree connecting the bacterial centers of mass. If we let $\mathcal{M}$ be a **minimum spanning tree** (MST) then longer edges in the tree will coincide with regions of the most negative Gaussian curvature. As any fragmentation of the aggregate must sever at least one edge of $\mathcal{M}$, we choose these edges as potential fragmentation locations. Given an edge e of $\mathcal{M}$, we define the plane $\mathcal{P}_e$ that bisects e and is normal to it. Thus $\mathcal{P}_e$ is normal to $\mathbf{x}-\mathbf{y}$ and passes through $\frac{1}{2}(\mathbf{x}-\mathbf{y})$, where $\mathbf{x}$ and $\mathbf{y}$ are the coordinates of the nodes of e. Supposing that we have chosen an edge e, then at any time point t we can compute the fragmentation force using equation (2.10), which we now denote as $F(e,t)$.

This procedure is illustrated in Figure 2.6. The red and blue dots are the centers of mass of each bacteria in the aggregate, and the gray lines are the edges of a minimum spanning tree connecting them. The highlighted edge near the center of the aggregate is an example candidate edge for fragmentation. The blue plane is the fragmentation plane that bisects this edge and is normal to it. The bacterial centers of mass are then color-coded according to which side of the MST they belong to, so that if the aggregate were to fragment at this edge, the red centers of mass would form one daughter aggregate and the blue centers of mass would form the other daughter.

Fragmentation of an Aggregate If the force $F(e,t)$ exceeds some threshold f_{crit} then the aggregate will fragment. We remove the edge e from the MST $\mathcal{M}$, which will result in two new trees. Each tree will become a new aggregate, which we construct by rotating and scaling the original bacterial centers of mass according to the shape and orientation of the ellipsoid $\mathcal{E}$ at time t. If $\mathbf{x}(0)$ are the coordinates of a bacterial center of mass at time $t = 0$, $\mathbf{R}(t)$ is the rotation specifying the orientation of $\mathcal{E}(t)$, and $\mathbf{a}(t)$ are the axes lengths of the ellipsoid, then the location of the center of mass at time t is

$$\mathbf{x}(t) = \mathbf{R}(t) \cdot \frac{\mathbf{a}(t)}{\mathbf{a}(0)}$$

where the division is taken element-wise. In this manner we can transform the centers of mass in each of the two new aggregates.

We have not yet specified how we wish to choose the edge e. Ultimately we will check all of the edges, but the order in which we do so matters. We use three different methods for comparison. In our simulations, the ellipsoid shape and orientation is cyclic in time, and hence it suffices to compute the fragmentation force over a single period T. Suppose the edges in the MST are sorted by their lengths at time $t = 0$. The **first edge** method gives the longest edge in the MST the chance to break before checking any other edges, so that if at some time t in T we find that $F(e_1, t) > f_{\text{crit}}$, then the aggregate fragments at e_1 at time t. Should $F(e_1, s)$ never exceed f_{crit}, then we check e_2 at all time points, and so on. This is the method we employed in our previous work [33], arguing that longer edges ought to be associated with surface regions possessing large negative Gaussian

curvature, which in turn would thus be more likely to break. The next method, **first time**, still gives preferential treatment to longer edges, but checks the force on each edge at time s before moving on to time $s + \Delta t$, so that a shorter edge e_j $(j > 1)$ has the opportunity to break at time s before any edge at time $s + \Delta t$. Lastly, the **global maximum** method computes $F(e, t)$ for all edges and all times in the discretization, and chooses the largest one, $F_{\max}$, which occurs at some edge d and time t. If $F_{\max} > f_{\text{crit}}$ then the aggregate fragments along edge d and at time t. This method gives no preferential treatment to either time or edge length.

Post-fragmentation density function

Given an aggregate and a fragmentation method, we can check for fragmentation as described above. If the aggregate fragments, we then have two new aggregates that we can submit to the same procedure. Eventually fragmentation will stop, either because all of the aggregates are reduced to singletons, which cannot fragment, or because all remaining aggregates experience a maximum fragmentation force below the critical force f_{crit}. We refer to this process as **exhaustive fragmentation**. Keeping track of the sizes of all intermediate flocs, we can then compute the density of fragmentations by the size of the parent aggregate. This in turn permits the construction of a post-fragmentation density function $\Gamma(k \mid m)$, which gives the probability of a fragmentation resulting in daughters of size k and $m - k$ given that an aggregate of size m fragments. We will construct $\Gamma(k \mid m)$ for each of the three fragmentation methods described above.

Model Parameters

The model parameters are described in Table 2.1. The DCM depends upon the initial axes lengths a_i of the ellipsoid, the velocity gradient $\mathbf{L}$, the matrix viscosity μ, the viscosity ratio λ, which is the ratio of the droplet viscosity over the matrix viscosity, and the interfacial tension Γ. The FCM depends upon the axes lengths $a_i(t)$ at each time point, the angular velocity of the ellipsoid $\boldsymbol{\omega}(t)$, and the velocity gradient $\mathbf{L}(t)$, which now also has a dependence on time due to the rotating frame of reference. In all of our simulations we set the external pressure p_0 to 0. Under this parameter regime, taking the external (matrix) fluid to be water, the Reynolds number is typically of order 10^{-2}, although adversarial choices of the length scale and velocity can give a Reynolds number

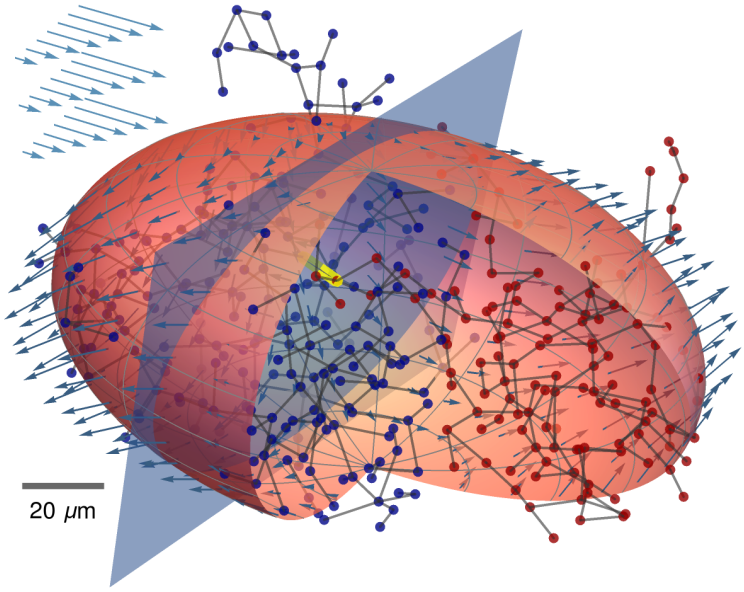


Figure 2.6: Mathematical representation of an aggregate. Bacterial centers of mass (blue and red points) are connected by a minimum spanning tree and sample force density vectors are shown on the ellipsoidal surface. An intersecting plane bisects the (yellow) highlighted edge of the MST. The centers of mass are color-coded (red and blue) to indicate the two daughter aggregates that would result from a fragmentation at this edge.

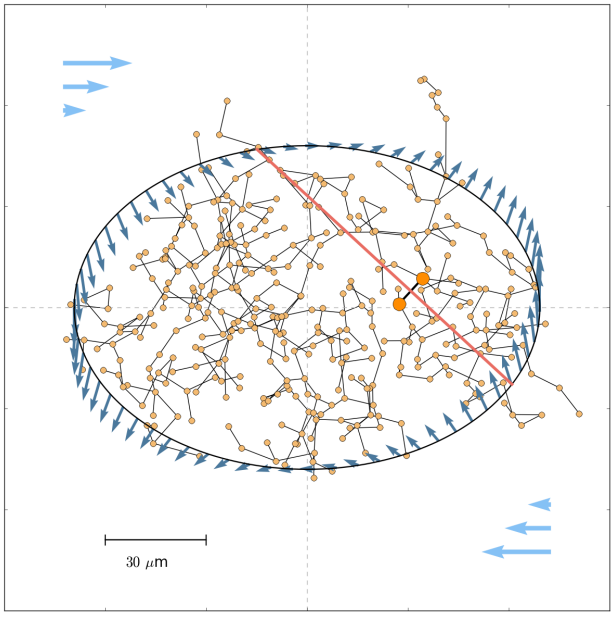


Figure 2.7: Mathematical representation of an aggregate. Bacterial centers of mass are connected by a minimum spanning tree and sample force density vectors are shown on the ellipsoidal surface. An intersecting plane bisects a sample edge of the MST. Force vectors are sample surface force densities.

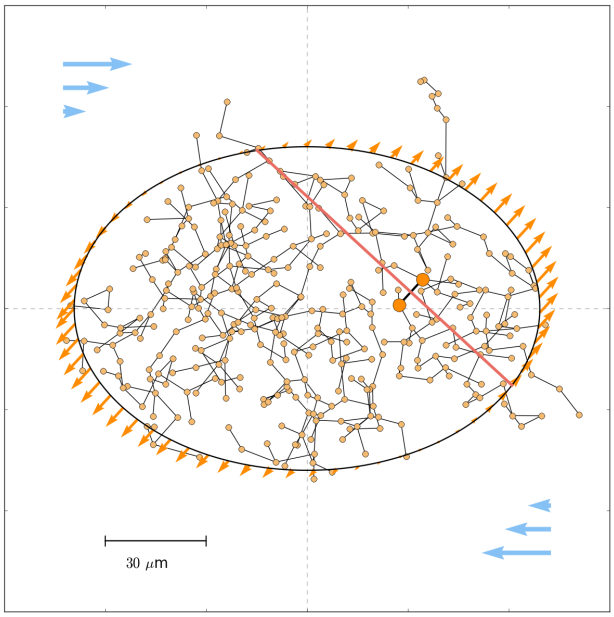
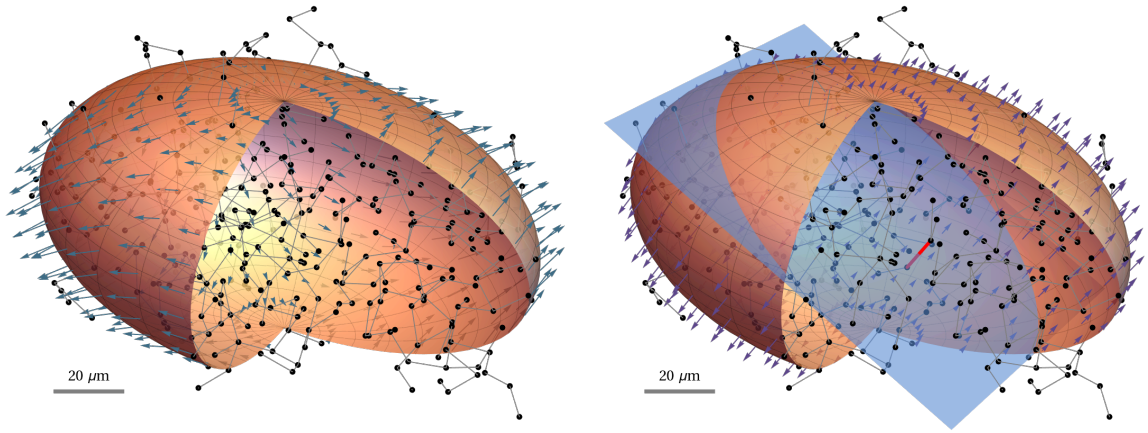


Figure 2.8: 2D Mathematical representation of an aggregate. Bacterial centers of mass are connected by a minimum spanning tree and sample force density vectors are shown on the ellipsoidal surface. An intersecting plane bisects a sample edge of the MST. Force vectors depict component of surface force density normal to the fragmentation plane.



(a) (b)

Figure 2.9: Mathematical representation of an aggregate, including ellipsoidal surface approximation, bacterial centers of mass, and a minimum spanning tree. Sample surface force vectors are shown in (a), and a sample fragmentation plane bisecting an arbitrary edge (red) along with normal components of the surface forces are shown in (b).

Symbol	Parameter	Model	Range	Units
$\mathbf{a}(t)$	axes lengths	D, F	1-1000$\times 10^{-6}$	m
$\boldsymbol{\omega}(t)$	angular velocity	F	0-100	1/s
$\mathbf{L}(t)$	velocity gradient	D, F	0-10	1/s
μ	matrix viscosity	D, F	8.95×10^{-4}	Pa s
λ	viscosity ratio	D	1-100	-
Γ	interfacial tension	D	$10^{-9} - 10^{-7}$	N / m

Table 2.1: Model parameters.

of order 1. Reducing the shear rate by an order of magnitude results in very little deformation of the ellipsoids, which effectively reproduces our earlier simulations of the fragmentation of solid ellipsoids. We use $\dot{\gamma} \sim 1\text{s}^{-1}$ to enforce some deformation, even though this increases the Reynolds number beyond a desirable order for Stokes' flow simulations.

We must make a further cautionary remark regarding the interpretability of the parameter values in our model. The viscosity ratio and interfacial tension parameters serve to induce a particular behavior in the system (oscillating deformation with tumbling, as we will discuss below), but the mechanisms physically responsible for this behavior are more likely viscoelastic. There are attractive and repulsive forces between the cells in the aggregates, for example, and the cohesion and retraction of the ellipsoidal shape is surely due to these rather than interfacial tension. We will later observe the qualitative dependence of our model's behavior on its parameters, and in so doing we must be careful not to overstate the physical interpretability of the specific values these parameters take. Their ranges were chosen to induce the oscillating behavior we expect to see, and should not be taken as representative of a physical regime of interest.

2.3 Results

2.3.1 Motion Under the DCM

Characteristic behaviors of the DCM An ellipsoid evolving according to the DCM follows one of three characteristic behaviors: it can (1) collapse smoothly to a steady-state orientation and shape (Fig 2.10a), (2) collapse while oscillating to a steady-state orientation (Fig 2.10b),

or (3) oscillate periodically (Fig 2.10c). The angular velocity of the oscillating collapse (Fig 2.10b) can often exhibit a sudden "flip" in which the direction of rotation of the ellipsoid changes. This occurs when two of the axes lengths are close in magnitude. In the remainder of the present work we restrict ourselves to the case of periodic tumbling (see Figure 2.10c), which we believe to be the most physically accurate for the case of microbial aggregates.

Futhermore, we expect it is unphysical for such an aggregate to become too elongated, and even if it were to, there is no reason to expect under such a circumstance that the model we employ here would accurately capture the physics responsible for such deformation. In such circumstances we might instead turn to a viscoelastic model of suspended colloids, such as in [42]. We therefore wish to restrict the maximum magnitude of axes length oscillations, which we can characterize with the parameter $d_{\max} = \max a_1(t)/\min a_1(t)$. The magnitude of the axes length oscillations depends primarily on the viscosity ratio and the shear rate, and we observe in practice that the other restrictions we impose on these parameters (Table (1)) result in bounding $d_{\max} < 2$.

Limiting viscosity In the oscillatory regime shown in Fig 2.10c, the behavior of the deforming ellipsoid approaches that of a solid ellipsoid. In simple shear defined by $d\mathbf{u}/dy = \dot{\gamma}$, the angle $\theta(t)$ in the xy plane of a solid ellipsoid is given ([43] c.f. [38]) by

$$\theta_{solid}(t) = -\arctan\frac{a_1}{a_2}\tan\left(\frac{2\pi t}{T}\right) \tag{2.12}$$

where

$$T = \frac{2\pi(a_1^2 + a_2^2)}{a_1 a_2 \dot{\gamma}} \tag{2.13}$$

is the period of the rotation. From this we can compute the angular velocity component ω_z as

$$\omega_{z,solid} = -\frac{2\pi}{T}\frac{a_1 a_2 \sec^2(2\pi t/T)}{a_2^2 + a_1^2\tan^2(2\pi t/T)}. \tag{2.14}$$

In the limit $\lambda \to \infty$, a fluid droplet becomes a solid, in which case we expect that the axes lengths will become constant and the angular velocity computed using equation (2.9) will approach that given in equation (2.14). This is indeed what we observe; as the viscosity ratio increases, the axis length oscillations decrease (Figure 2.11a) and the angular velocity converges to that of a solid ellipsoid (Figure 2.11b).

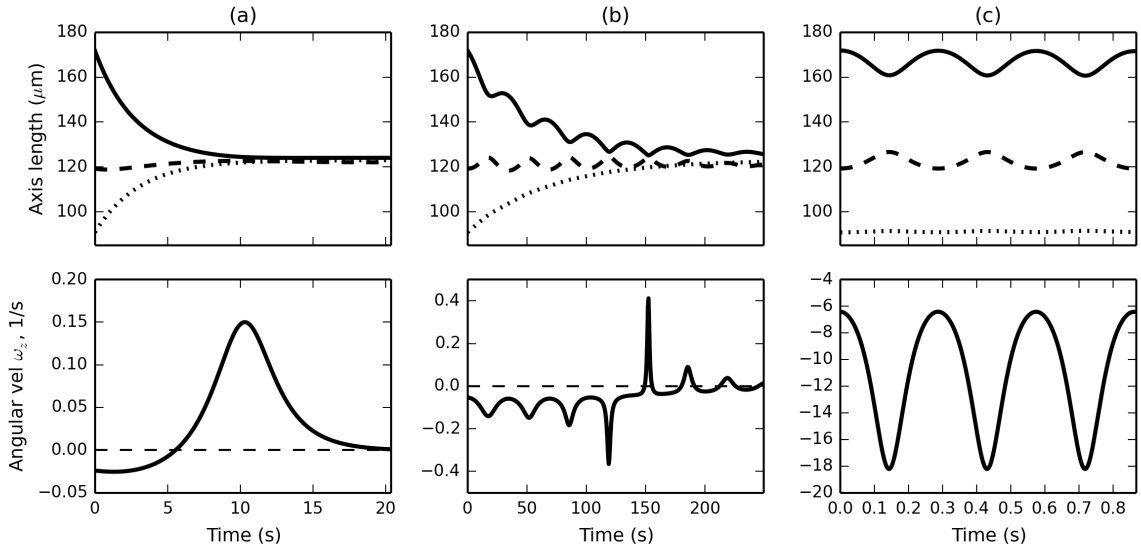


Figure 2.10: Characteristic behaviors of ellipsoids evolving in the DCM: (a) collapse (Ca $\sim$ .294), (b) oscillating collapse (Ca $\sim$ 13.2), and (c) periodic tumbling (Ca $\sim$ 3441). Here Ca is the capillary number, defined by $\text{Ca} = \mu V/\Gamma$, where μ is the dynamic viscosity of water, $V \sim \|\mathbf{a}\|\dot{\gamma}$ is a characteristic velocity, and Γ is the interfacial tension.

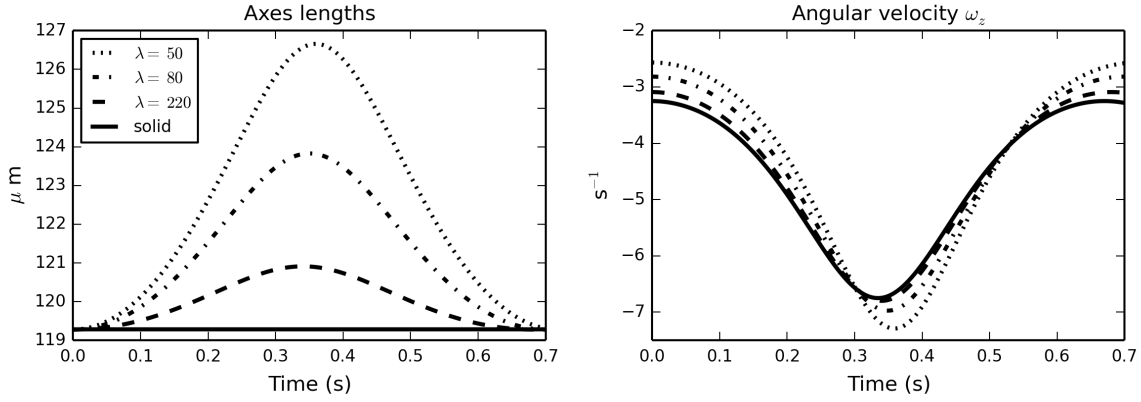


Figure 2.11: Asymptotic behavior of the DCM as $\lambda \to \infty$ (dashed lines) compared to the behavior of a solid ellipsoid with angular velocity given by equation (2.14) (solid line). Left: second axis length (a_2) over time, right: angular velocity component ω_z over time.

2.3.2 Fragmentation force

Qualitative behavior Figure 2.2 shows the evolution in time of a generic ellipsoid at four time-points, including the surface force density as well as the component of this density acting normal to a sample fragmentation plane. This ellipsoid is undergoing periodic tumbling, as in Figure 2.10c, with mild deformation. In the first frame, at the initial time, we observe that the surface force density points both outwards and inwards. This feature is responsible for the fact that at the third time point, when the angular velocity is maximized which in turn causes the surface force density magnitudes to be maximized, we nevertheless observe a net fragmentation force of 0. The maximum fragmentation force is observed at the second time-point, when all of the force vectors act against the plane, and the minimum, which is negative, occurs at the fourth time point, when all of the surface force vectors push into the plane.

Parameter dependence Intersect a generic ellipsoid $\mathcal{E}$ with a plane P defined by the normal $\mathbf{n}_p = (1, 0, 0)$ and interior point $\mathbf{x}_p = (0, 0, 0)$; i.e., a plane in the yz plane passing through the origin and normal to the longest major axis of $\mathcal{E}$. We first explore the dependence of the fragmentation force, equation (2.10), on the system parameters. We compute the maximum fragmentation force as a function of the shear rate $\dot{\gamma}$, the viscosity ratio λ (which we vary while holding the matrix viscosity μ constant, changing only μ^*), and the interfacial tension Γ. As can be seen in Figure 2.12a, the fragmentation force increases with the shear rate. The shear rate appears directly in the computation of the surface force density in equation (2.3), and indirectly as it affects the angular velocity $\boldsymbol{\omega}$. At higher shear rates there is a greater dependence of f_{max} on the viscosity ratio, and its dependence on λ is non-linear, changing more for smaller values of λ while being constant at higher values of λ. The dependence of f_{max} on Γ and λ is shown in Figure 2.12b. Again, f_{max} increases with λ; in addition, it can be seen to decrease with Γ. Neither of these terms appear directly in the force equation (2.3), and so their influence on f_{max} manifests through their role in shaping the motion and deformation of the ellipsoid as in equation (2.2).

Maximizing fragmentation force as a function of plane location We next consider

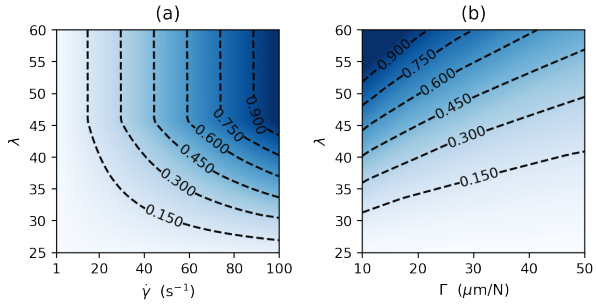


Figure 2.12: Maximum normalized fragmentation force experienced by a sample ellipsoid as a function of the shear rate $\dot{\gamma}$ and the viscosity ratio λ (a) and the interfacial tension Γ and the viscosity ratio (b). In (a), $\Gamma = 40 \times 10^{-6}$ N/m, and in (b) $\dot{\gamma} = 10$ m/s.

the fragmentation force as a function of time and position of the intersecting plane. We construct the ellipsoid as above, except that now we will slide the intersecting plane along the x axis. These results are shown in Figure 2.13. The x axis corresponds to the position of the interior point on the intersecting plane, so that at a point x on this axis, the intersecting plane is defined by normal $\mathbf{n}_p = (1, 0, 0)$ and $\mathbf{x}_p = (x, 0, 0)$. The y axis corresponds to time. The fragmentation force is anti-symmetric about its horizontal center, which corresponds to the point in time at which the ellipsoid has rotated through an angle of $\pi/2$. Past this point, the symmetry of the system results in the surface forces being equal in magnitude but opposite in sign. As the plane slides along the x axis to the midpoint, the fragmentation force increases, and then decreases again as the plane moves from the center to the other end; again due to the symmetries of the system.

2.3.3 Aggregate Fragmentation

Figure 2.14 shows the results of exhaustive fragmentation on the initial set of 39 aggregates for each of the three fragmentation methods. Each subplot shows a two dimensional histogram of the normalized frequency of fragmentations, in which the horizontal axis corresponds to the size of the parent aggregate, and the vertical axis to the ratio of the size of a daughter aggregate to its parent. Because a fragmentation of an aggregate of size m into a daughter of size k also necessarily gives rise to a daughter of size $m - k$, the plots are symmetric about the horizontal line $r = 0.5$.

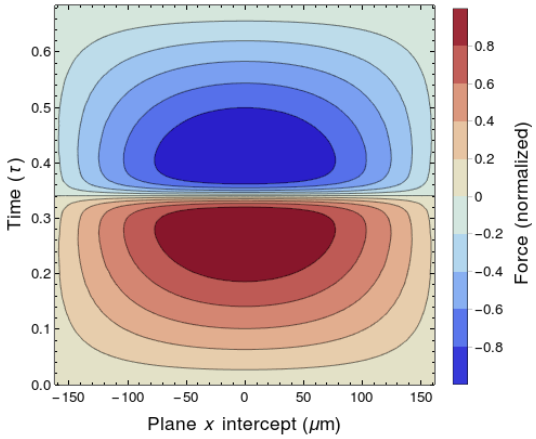


Figure 2.13: Normalized fragmentation force with respect to a plane normal to $(1, 0, 0)$ and intersecting a sample ellipsoid at $\mathbf{x} = (x, 0, 0)$ (horizontal axis) over time (vertical axis). $\mathbf{a}_0 = (180, 140, 100)$ μm.

The probabilities are normalized by parent size, so that the entry corresponding to $x = m$ on the horizontal axis and $r = k$ on the vertical axis indicates the probability that an aggregate of size m fragments into daughters of size k and $m - k$, given that the parent is of size m and fragments.

Figure 2.14a shows the results for the **first edge** fragmentation method. Recall that this method computes the fragmentation force for each edge, in order of length, over the entire period of motion before proceeding to the next edge. As the parent size increases, erosion quickly becomes the dominant fragmentation mechanism. This can be seen by the larger probabilities along the vertical boundaries of the plot: parents fragment into one small and one large daughter. This is the same pattern we observed in our original work on this problem, in which we used the first edge method and treated the aggregates as non-deformable solids [33].

Figure 2.14b shows the results for the **first time** fragmentation method, in which the fragmentation force is computed at each time point for each edge before moving on to the next longest edge. We still observe a tendency towards erosion as parent size increases, but it is less dominant than in the first edge method, which is to say that we observe an increase in the frequency of fragmentations into more evenly-sized daughters. Finally, Figure 2.14c shows the post-fragmentation density function for the **global maximum** method, which computes the fragmentation force at all times for all edges and chooses the largest one. This method shows a significant shift towards splitting in comparison to the other two methods, although erosion often still dominates.

We can make sense of these results as follows. The bacterial centers of mass tend be clustered in a central area, so that the density of these points decreases with distance from the origin. A minimum spanning tree taken on such a such a set of points is likely to have its longest edges towards the periphery of the aggregate. These edges are in turn more likely to connect singletons or smaller aggregates to the parent aggregate, as opposed to more central, shorter edges, which will tend to separate the tree into two more evenly sized subtrees. An example of this can be observed in Figure 2.6: the relatively short highlighted edge splits the tree into two subtrees of comparable size (red and blue centers of mass) and is buried near the center of the aggregate.

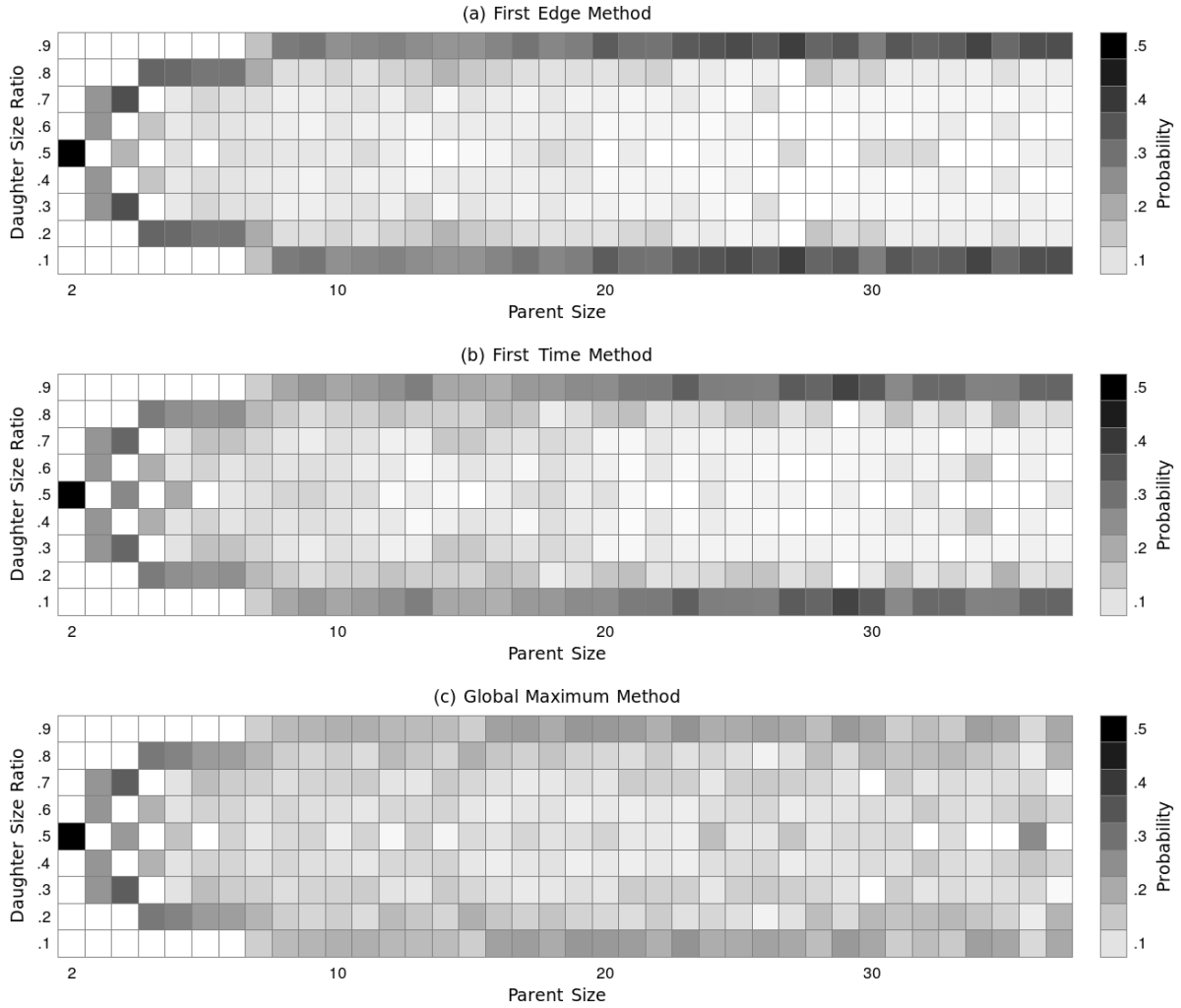


Figure 2.14: Post-fragmentation density functions for (a) the **first edge** method, (b) the **first time** method, and (c) the **global maximum** method. The horizontal axis corresponds to the size of the fragmenting aggregate, and the vertical axis to the ratio of daughter size to mother size.

2.4 Conclusions and Future Plans

We have presented a method to check for fragmentation in free-floating aggregates in simple shear flow which combines a model for the forces on the surface an ellipsoid with one for the deformation of an ellipsoid in flow. We then used this method in a simulation of exhaustive fragmentation on an initial dataset based on images of 39 microbial aggregates. We compared three different methods for checking for fragmentation, one which we have used previously in a simpler fragmentation simulation. The post-fragmentation density functions we obtain for each method are different than the normal and log-normal distributions which are widespread in the literature. In particular, the functions exhibit a range of behaviors from almost pure erosion in the case of the first edge method to an erosion/splitting mix in the case of the global maximum method. Even in the latter case, erosion remains an important fragmentation mechanism. These results can inform the choice of fragmentation kernels in population-balance models, primarily by suggesting a more important role for erosion than is generally assumed.

With this hybrid methodology, we are now ready to proceed with upscaling these results to a population level model as well as comparing the predictions with experimental size-structured population data.

Acknowledgments

EPK is supported by the Interdisciplinary Quantitative Biology Program at the BioFrontiers Institute, University of Colorado Boulder (NSF IGERT 1144807) and by an NSF GRFP (DGE 1144083). This work was supported in part by grant NSF-DMS 1225878 to DMB. We would like to thank Charles Tucker III, Eric Wetzel, and Nancy Jackson for making their code available to us and for helpful discussions on the behavior of their model, and Dr. John Younger for helpful discussions on an application of this work to microbial flocculation.

Chapter 3

One-Pass Sparsified Gaussian Mixtures

This chapter is adapted from [2] (under review at the time of publication). Research was supported in part by NSF GRFP DGE 1144083.

We present a one-pass sparsified Gaussian mixture model (SGMM). Given P-dimensional datapoints $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, the model fits K Gaussian distributions to $\mathcal{X}$ and (softly) classifies each $\mathbf{x}_i$ to these clusters. After paying an up-front cost of $\mathcal{O}(NP \log P)$ to precondition the data, we subsample Q entries of each datapoint and discard the full P-dimensional data. SGMM operates in $\mathcal{O}(KNQ)$ time for diagonal or spherical covariances, independent of P, while estimating the model parameters $\boldsymbol{\theta}$ in the full P-dimensional space, making it one-pass and hence suitable for streaming data. We derive the maximum likelihood estimators for $\boldsymbol{\theta}$ in the sparsified regime, demonstrate clustering on synthetic and real data, and show that SGMM is faster than GMM while preserving accuracy.

3.1 Introduction

When performing clustering analysis on high-dimensional (P features), high-volume (N samples) data, it is common to employ simple clustering schemes like k-means and k-nearest-neighbors, particularly during data exploration and feature engineering, because these techniques are fast and return informative results [44]. Often each datapoint $\mathbf{x}_i \in \mathbb{R}^P$ will be seen only once and must then be discarded, necessitating **one-pass** algorithms [45]. Futher, the latent dimesion P may be prohibitively large or the rate of data acquisition may be too high to permit analysis on the full

data.

We present a clustering algorithm suitable for this regime: the sparsified Gaussian mixture model (SGMM), building on our previous work in which we developed sparsified k-means clustering [46]. GMM, in particular when using diagonal or spherical covariances, is a natural extension of k-means: it increases generalizability by taking into account cluster size and covariance and by performing soft clustering, while still being relatively inexpensive to compute [47].

SGMM works on compressed data, such that the computation and storage costs are measured in $Q \ll P$, and yet is one-pass, meaning that the model parameters are estimated in the full P-dimensional space. These requirements are seemingly orthogonal to each other; we are able to provide both by a careful choice of how we compress the data, which we do using a **sketch** $\mathbf{R}_i^T \mathbf{x}_i$ of size $Q \ll P$. Our sketching scheme is motivated by the Johnson-Lindenstrauss lemma [48], which states that certain random projections into lower dimensions preserve pairwise distances to within a small error ε with high probability. In particular, these embeddings can be computed efficiently in $\mathcal{O}(NP \log P)$ time [49], and the data are recoverable from the embeddings when they are sparse in some basis [50]. The idea is to project the data into a lower dimension and perform analyses there, where it is cheap to do so. A variety of approaches have been proposed to this end [51, 52, 53, 54, 55], including applications of sketching to Gaussian Mixture models [56, 57]. Our work is distinct from these approaches by virtue of being one-pass.

In general, such compressive approaches are two-pass, meaning that access to the full data is required to estimate statistics in the original space, such as the sample mean. The contribution of our method is that it is compressive **and** one-pass, meaning that we estimate statistics in the full P-dimensional space using only Q-dimensional sketches of the data.

The chapter is organized as follows. In Section 3.2 we discuss the Johnson-Lindenstrauss lemma with an emphasis on applications to dimensionality reduction for clustering. We also discuss the concept of one-pass algorithms and introduce our one-pass subsampling scheme. In Section 3.3 we introduce mixture models in general and Gaussian mixtures in particular, as well as the Expectation-Maximization method used to estimate model parameters. In Section 3.4 we apply

our sparsification scheme to Gaussian mixtures to derive our main result, the Sparsified Gaussian mixture model (SGMM). We derive the maximum likelihood estimators for the Expectation-Maximization algorithm applied to SGMM, and find the computational complexity of the overall algorithm. In section 3.5 we present some results from simulations using SGMM on real and synthetic data. Finally, in Section 3.6 we summarize our results and introduce future topics of research.

3.2 Dimensionality Reduction and the Johnson-Lindenstrauss Lemma

3.2.1 The Johnson-Lindenstrauss Lemma

The Johnson-Lindenstrauss (JL) lemma ([48] cf. [58]) is:

Theorem 1 (Johnson-Lindenstrauss)**.** *For any $0 < \varepsilon < 1$ and any integer N, let Q be a positive integer such that*

$$Q \geq 4(\varepsilon^2/2 - \varepsilon^3/3)^{-1} \log N. \tag{3.1}$$

Then for any set $\mathcal{X}$ of N points in $\mathbb{R}^P$, there is a map $\Phi : \mathbb{R}^P \to \mathbb{R}^Q$ such that for all $u, v \in \mathcal{X}$,

$$(1 - \varepsilon) \leq \frac{\|\Phi(u - v)\|_2^2}{\|u - v\|_2^2} \leq (1 + \varepsilon). \tag{3.2}$$

The map Φ embeds a set of N observations $\mathcal{X}$ from P-dimensional space into Q-dimensional space without distorting any pairwise relative distances more than ϵ, where Q is bounded below by a constant of size $\mathcal{O}(\varepsilon^{-2} \log N)$. Note that this bound is independent of P, the original dimension of the data.

Suppose that we wish to perform some sort of computation on $\mathcal{X}$ that depends on pairwise distances between points $u, v \in \mathcal{X}$ rather than on the component entries $\{u_i\}_{i=1}^P$, for example, nearest-neighbor searching. In such a case we could apply JL to embed $\mathcal{X}$ into Q-dimensional space and then perform our analyses on a potentially much smaller dataset. This principle, known as **metric embedding**, has been well-studied [59, 60, 61, 62], including applications to the specific task of approximate nearest neighbor searches [63, 64].

We have yet to discuss how expensive it is to find and apply the embedding f in Theorem 1. In the original proof ([48] cf. [49]) f is represented as a $Q \times P$ matrix Φ. The rows of Φ are chosen iteratively to be orthogonal random unit vectors. Obtaining and applying such a transform has a polynomial cost, which is the bottleneck in fast computations like nearest-neighbor searches. Subsequent analyses improved upon this by finding simpler structure and introducing some sparsity into Φ [51, 58, 63, 65].

It was shown that there is a hard lower-bound on Q of order $\mathcal{O}(1/\varepsilon)$ [66], motivating efforts to introduce further sparsity into Φ. The central development in this line of research for our line of work is the Fast Johnson-Lindenstrauss Transform (FJLT) [49], which speeds up the application of the projection Φ by way of the **Heisenberg principle**, stating that a signal and its spectrum cannot both be concentrated.

3.2.2 The Fast Johnson-Lindenstrauss Transform

The FJLT [49] defines the transform Φ as $\Phi = \mathbf{RHD}$, where:

- $\mathbf{R}$ is a $Q \times P$ sparse normal distribution matrix (details omitted; see [49])
- $\mathbf{H}$ is a $P \times P$ Hadamard matrix
- $\mathbf{D}$ is a $P \times P$ diagonal matrix with entries drawn from ± 1 with equal probability.

Here $\mathbf{R}$ and $\mathbf{D}$ are random and $\mathbf{H}$ is deterministic. The application of $\mathbf{R}$ serves to preserve Q bits from P, while $\mathbf{HD}$ is a preconditioning step. If we were to apply $\mathbf{R}$ without preconditioning, we are relying on $\mathbf{R}$ to, by chance, extract the important information from $\mathbf{x}_i$. In the extreme case where $\mathbf{x}_i = \mathbf{e}_l$ (an elementary unit vector), then $P(\mathbf{R}\mathbf{x}_i) = 0 \to 1$ in the limit $n \to \infty$. Preconditioning with $\mathbf{H}$ distributes the information in $\mathbf{x}_i$. In particular, $\mathbf{H}\mathbf{e}_l$ will be smooth over its entries as the image of a peaked distribution mapped to the Fourier domain. Conversely, however, if $\mathbf{x}_i$ is already smooth then its image in Fourier space might be peaked, in which case we risk losing the signal through subsampling with $\mathbf{R}$. The matrix $\mathbf{D}$ serves to scramble $\mathbf{x}_i$ so that is it probabilistically

not smooth, ensuring with high probability that the application of $\mathbf{H}$ does not give a peaked distribution.

Using this compression scheme has the following bound on norm distortion:

Theorem 2 (FJLT for ℓ_2 cf. Ailon and Chazelle [49])**.** *Let $X = \{\mathbf{x}_i\}_{i=1}^N$ with $\mathbf{x}_i \in \mathbb{R}^P$, choose $\varepsilon < 1$, and let Φ be a FJLT as described above. Then with probability at least 2/3, the following holds. For all $\mathbf{x}_i \in X$:*

$$c\left(\frac{1-\varepsilon}{\varepsilon^2}\right) \leq \frac{\|\Phi\mathbf{x}_i\|_2}{\|\mathbf{x}_i\|_2} \leq c\left(\frac{1+\varepsilon}{\varepsilon^2}\right) \tag{3.3}$$

where c is a global constant that is easy to compute and can be can absorbed into Φ.

There are a wealth of results and guarantees about this type of preconditioning. The specific example we discussed above comes with several such bounds discussed in detail in [49]. Thorough monographs on the broader topic of **randomized matricies** give many more results [67, 68]. Previous work in our group establishes guarantees specific to our subsampling strategy [46], which we introduce in the following section. For our purposes here it is sufficient to note that this type of preconditioning and random subsampling is consistent with the JL bounds probabilistically, and can be applied in $\mathcal{O}(NP \log P)$ operations.

We now discuss how we modify this approach to sparsification for use with one-pass algorithms.

3.2.3 Sketching for One-Pass Algorithms

We will project $\mathbf{x}_i$ into a lower dimension by keeping $Q \ll P$ components chosen unformly at random. Before doing so we precondition the data using a random orthonormal system (ROS):

$$\mathbf{x}_i = \mathbf{HD}\mathbf{x}_i^{raw} \tag{3.4}$$

where $\mathbf{D}$ is diagonal with entries ± 1 chosen uniformly at random and $\mathbf{H}$ is a discrete cosine transform matrix[1] . The ROS transformation ensures that, with high probability, the magnitudes of the entries

[1] other choices include Hadamard or Fourier

of $\mathbf{x}_i$ are relatively close to each other [69, 49], minimizing the risk of "missing" the information in the vector when subsampling. The ROS can be applied and inverted in $\mathcal{O}(NP \log P)$ time, which is the dominant cost in our algorithm for small enough sketches. A detailed discussion of convergence properties and bounds of the ROS can be found in [46]. Henceforth, when we write $\mathbf{x}_i$ we assume the data have been preconditioned.

Following the preconditioning, we subsample $Q \ll P$ entries chosen uniformly at random from $\mathbf{x}_i$. This operation can be represented by the product $\mathbf{R}_i^T \mathbf{x}_i$ where $\mathbf{R}_i \in \mathbb{R}^{P \times Q}$ is sparse, with $\mathbf{R}_i(p, q) = 1$ if we are keeping the pth feature of $\mathbf{x}_i$ and storing it in the qth dimension of the sparsified vector, and 0 otherwise. Thus $\mathbf{R}_i^T \mathbf{x}_i \in \mathbb{R}^Q$ are the entries we preserve from $\mathbf{x}_i$. In practice we store only the Q entries of $\mathbf{x}_i$ that the subsampling picks out as well as the indices specifying which entries were preserved (the indices of the Q non-zero rows of $\mathbf{R}_i$), though it will facilitate our exposition to write quantities like $\mathbf{R}_i \mathbf{R}_i^T \mathbf{x}_i \in \mathbb{R}^P$. Crucially, $\mathbf{R}_i$ is resampled for each $\mathbf{x}_i$. This fact is what enables the method to estimate statistics in P dimensions with a single pass through the data (i.e., one-pass).

3.3 Mixture Models

We now describe the modeling framework, beginning with a general mixture model [47]. Assume there are K components and that each datapoint $\mathbf{x}_i$ belongs to one of them, indicated by the hidden variable $z_i \in \{1, 2, \dots K\}$. A **mixture model** [47] is fully specified by the component distributions $p_k(\mathbf{x}_i \mid \boldsymbol{\theta}) = p(\mathbf{x}_i \mid z_i = k, \boldsymbol{\theta})$, the component weights $\boldsymbol{\pi} = \{\pi_k\}_{k=1}^K$ with $\sum \pi_k = 1$, and the parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_k\}_{k=1}^K$. The distribution for x_i is given by

$$p(\mathbf{x}_i \mid \boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k p_k(\mathbf{x}_i \mid \boldsymbol{\theta}). \tag{3.5}$$

For a mixture of Gaussians, $\boldsymbol{\theta}_k = \{\boldsymbol{\mu}_k, \mathbf{S}_k\}$ where $\boldsymbol{\mu}_k \in \mathbb{R}^P$ is the mean and $\mathbf{S}_k \in \mathbb{R}^{P \times P}$ is the covariance of the kth cluster, and $p(\mathbf{x}_i \mid z_i = k, \boldsymbol{\theta})$ is given by

$$p_k(\mathbf{x}_i \mid \boldsymbol{\theta}) = \frac{1}{(2\pi)^{P/2}} \frac{1}{|\mathbf{S}_k|^{1/2}} \exp\left(-\frac{1}{2} D_{\boldsymbol{\theta}}(\mathbf{x}_i)\right) \tag{3.6}$$

where

$$D_{\boldsymbol{\theta}_k}(\mathbf{x}_i) = \big(\mathbf{x}_i - \boldsymbol{\mu}_k\big)^T \boldsymbol{\Lambda}_k \big(\mathbf{x}_i - \boldsymbol{\mu}_k\big) \tag{3.7}$$

is the squared Mahalanobis distance and $\boldsymbol{\Lambda}_k = \mathbf{S}_k^{-1}$ is the kth precision matrix.

The goal is to simultaneously estimate the parameters $\boldsymbol{\theta}$, the weights $\boldsymbol{\pi}$, and the cluster assignments z_i, which we do using the Expectation-Maximization algorithm.

3.3.1 Expectation Maximization

Suppose we have observed data x, a realization of a random variable X, with probability distribution function $p_X(x;\theta)$. Note that by x we mean the full dataset, so that if, for example, x is a set of N measurements $\{y_i\}_{i=1}^N$ drawn IID from some distribution $p_Y(y;\theta)$ then

$$p_X(x;\theta) = \prod_{i=1}^{N} p_Y(y;\theta) \tag{3.8}$$

$$= L(\theta; X) \tag{3.9}$$

is the likelihood of x. Following the development in Appendix 1 of [70], we choose to use p_X and let x be the set of samples, rather than a single observation within the dataset, because this choice simplifies notation. At later times we will want p_X to be the likelihood of a mixture model and a partially-observed Bayesian network.

In this context, the goal of maximum likelihood estimation is to find estimators $\widehat{\theta}$ for parameters θ such that $\widehat{\theta}$ maximize p_X:

$$\widehat{\theta} = \arg\max_{\theta \in \Theta} p_X(x;\theta). \tag{3.10}$$

For many distributions there exist analytic expressions for $\widehat{\theta}$. Suppose now, however, that we introduce a **hidden** random variable Z with joint distribution p_{XZ}

$$p_X(x;\theta) = \int_Z p_{XZ}(x, z; \theta) dz. \tag{3.11}$$

Z is also sometimes called a **latent** variable or **missing data**. For our purposes Z will be the class labels for X in the case of a mixture model, as well as the unobserved or hidden nodes in

a partially-observed realization of a Bayesian network. These are only two examples of a wide variety of useful choices for Z. Since we do not observe Z we resort to seeking the maximizer of the marginal distribution:

$$\widehat{\theta} = \arg\max_{\theta \in \Theta} \int_Z p_{XZ}(x, z; \theta) dz. \tag{3.12}$$

This quantity, however, can be prohibitively difficult to compute. In such cases it is common that the marginal would be easy to compute given either Z or θ.

The Expectation-Maximization (EM) method [71] proposes a solution to this by leveraging the fact that the problem would be solvable given either Z or θ. In Expectation-Maximization, one of these quantities is held constant and an update is obtained for the other in an iterative manner. The E step estimates the hidden variables Z given θ^t, the current estimates for the parameters θ at step t. The M step then finds the next parameter iterate θ^{t+1} using the esimates for Z from the E-step. Expectation-Maximization has several desirable properties: the likelihood $L(\boldsymbol{\theta}; X)$ is non-decreasing in t, and the process is guaranteed to converge to a critical point of $L(\boldsymbol{\theta}; X)$ [72, 73, 74, 75], which in practice is almost always a maximum (as opposed to a saddlepoint) [70].

3.3.1.1 The Auxiliary Function

Expectation-Maximization does not use the likelihood function $p_X(x; \theta)$ directly. Instead, define the **auxiliary function**

$$\begin{aligned} Q\left(\theta, \theta^{t-1}\right) &= E_{Z|X;\theta^{t-1}}\left[\log p_{XZ}(x, z; \theta)\right] &(3.13)\\ &= \int_Z \log p_{XZ}(x, z; \theta)\, p_{Z|X}\left(z|x\,;\, \theta^{t-1}\right) dz. &(3.14)\end{aligned}$$

This corresponds to the expectation of the log likelihood taken with respect to the conditional density $p_{Z|X}$, which is the density of Z conditioned on the observed data x using the parameter estimate θ^{t-1}. Although it is not necessarily obvious from this formulation, equation (3.13) is often more tractable to compute than the original formulation in equation (3.12).

Furthermore, we typically do not need to evaluate Q in equation (3.13) directly at all. Instead,

we only need to compute enough to find

$$\theta^t = \arg\max_{\theta \in \Theta} Q(\theta, \theta^{t-1}). \tag{3.15}$$

In many cases, notably when p_X is an exponential family distribution, evaluating the relevant parts of Q in equation (3.13) amounts to finding a sum of factored sufficient statistics with simple, analytic expressions [72, 73, 74]. We will see an example of this in the case of Gaussian mixtures.

Expectation-Maximization proceeds in two steps, the E step and the M step. In the E step, the necessary parts of Q are computed using current estmimates for θ and Z. In the M step, a new update to θ is found by maximizing Q with respect to θ. The choice of Q is motivated intuitively by the idea that we can more easily compute quantities conditioned on θ and x, but in order for Expectation-Maximization to work as intended, the sequence θ^t must also maximize the likelihood $p_X(x; \theta)$. This is indeed the case; for a thorough derivation see [76].

3.3.1.2 E and M steps

E Step The E step is to compute the necessary components in the auxiliary function at time $t-1$; i.e., evaluate $Q(\theta, \theta^{t-1})$. We will see two different forms of Q. One is for a Gaussian mixture mode, and the other for a Bayesian network. In both cases the distributions factorize and the E step reduces to a closed-form evaluation of a sum of sufficient statistics.

M Step The M step is to find θ to optimize Q found in the E step:

$$\theta^t = \arg\max_{\theta \in \Theta}(Q(\theta, \theta^{t-1})). \tag{3.16}$$

In practice it turns out to be sufficient to set $dQ/d\theta = 0$ and find the critical point. Sometimes we will also need to find additional parameters that are not specified in the vector θ; for instance, in the case of Gaussian mixtures we will seek the weights $\boldsymbol{\pi} = \{\pi_k\}_{k=1}^K$, where K is the number of components in the mixture. This is also done during the M step. The EM algorithm is presented in Algorithm 3.1.

Algorithm 3.1: Expectation-Maximization

Inputs : x, observed data.
θ_0, initial parameter estimate.
$\Delta\theta_{tol}$, convergence tolerance.
t_{max}, maximum iterations.
Output: θ^t

initialize $\Delta\theta, t$;
while $(\Delta\theta > \Delta\theta_{tol})$ **and** $(t \leq t_{max})$ **do**
  (E step): compute $Q(\theta, \theta^{t-1})$ or sufficient statistics;
  (M step): $\theta^t \leftarrow \arg\max_\theta Q(\theta, \theta^{t-1})$;
  $t \leftarrow t + 1$;
  $\Delta\theta \leftarrow \left\|\theta^t - \theta^{t-1}\right\|$;
end while

3.3.2 The EM Algorithm for Gaussian Mixtures

We now specialize the EM algorithm to the case of Gaussian Mixture Models. The log likelihood for data $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ under the mixture distribution given in equation (3.5) is

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} p_k(\mathbf{x}_i \mid \boldsymbol{\theta}_k) \right). \tag{3.17}$$

In the case of GMM's (as well as in many others) it is intractable to find the maximum likelihood estimators (MLE's) for $\boldsymbol{\theta}$ because of the hidden $\mathbf{z} = \{z_i\}_{i=1}^N$. Expectation-Maximization finds a local optimum by iteratively holding one of the unknown quantities ($\boldsymbol{\theta}$ or $\mathbf{z}$) fixed and solving for the other. At each iteration we obtain a new estimate $\{\boldsymbol{\theta}^t, \boldsymbol{\pi}^t\}$ computed from the previous estimate $\{\boldsymbol{\theta}^{t-1}, \boldsymbol{\pi}^{t-1}\}$. Specifically, define the auxiliary function

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1}) = E\left[\ell_c(\boldsymbol{\theta}) \mid \mathcal{X}, \boldsymbol{\theta}^{t-1}\right] \tag{3.18}$$

where

$$\ell_c(\boldsymbol{\theta}) = \sum_{i,k} \log p\left(\mathbf{x}_i, z_i = k \mid \boldsymbol{\theta}_k\right) \tag{3.19}$$

is the **complete data log likelihood** and $\mathcal{X}$ is the dataset.

The E step is then to compute the expected sufficient statistics in Q for $\boldsymbol{\theta}$, which is equivalent to finding the **responsibility** $r_{ik} = p(z_i = k \mid \mathbf{x}_i, \boldsymbol{\theta}^{t-1})$ for each datapoint $\mathbf{x}_i$ and component k:

$$r_{ik} = \frac{\pi_k p_k(\mathbf{x}_i \mid \boldsymbol{\theta}^{t-1})}{\sum_{j=1}^{K} \pi_j p_j\left(\mathbf{x}_i \mid \boldsymbol{\theta}^{t-1}\right)}. \tag{3.20}$$

The auxiliary function in equation (3.18) can then be expressed in terms of the responsibility as

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1}) = \sum_{i,k} r_{ik} \log \left[\pi_k p_k(\mathbf{x}_i \mid \boldsymbol{\theta})\right]. \tag{3.21}$$

The M step is to obtain the next iterate $\{\boldsymbol{\theta}^t, \boldsymbol{\pi}^t\}$ by optimizing Q:

$$\{\boldsymbol{\theta}^t, \boldsymbol{\pi}^t\} = \text{argmax}_{\boldsymbol{\theta},\boldsymbol{\pi}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1}). \tag{3.22}$$

For a mixture of Gaussians, Q is optimized by the the maximum likelihood estimators:

$$\widehat{\pi}_k = \frac{r_k}{N} \tag{3.23}$$

$$\widehat{\boldsymbol{\mu}}_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}} \tag{3.24}$$

$$\widehat{\mathbf{S}}_k = \frac{\sum_i r_{ik}(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_k)^T}{\sum_i r_{ik}} \tag{3.25}$$

The E and M steps are repeated until (guaranteed) convergence to a local maximum or saddle point of Q.

3.4 Sparsified Gaussian Mixture Models

We now derive the maximum likelihood estimators for π_k and $\boldsymbol{\theta}_k$ under sparsification, the main theoretical result of this chapter. To do so we will need some specialized results from matrix calculus. Collections of similar results can be found in appendix A.4.1 of [77] as well as in the reference guide [78]. The latter is an exceptionally useful reference.

3.4.1 Some Specialized Results from Matrix Calculus

The parameters $\boldsymbol{\theta}$ we seek to estimate are P-dimensional: $\boldsymbol{\mu}_k \in \mathbb{R}^P$ and $\mathbf{S}_k \in \mathbb{R}^{P\times P}$, and hence to find the maximum likelihood estimators for them we need to take derivates in P dimensions. We can choose to compute these as P-dimensional operators acting on sparse arrays of the form $\mathbf{R}_i{}^T\mathbf{R}_i(\cdot)\mathbf{R}_i{}^T\mathbf{R}_i \in \mathbb{R}^{P\times P}$ or on dense arrays of the form $\mathbf{R}_i{}^T(\cdot)\mathbf{R}_i \in \mathbb{R}^{Q\times Q}$. Because we only need to compute gradients of scalar functions, we can do so element-wise and rewrite the result in either form, depending on which is most convenient.

Definition 3 (Mask indicator)**.** *Let* $\mathbf{R}_i$ *be a sampling projection. The* ***mask indicator*** $\delta^{\mathcal{R}_i} : \{1, 2, \dots P\} \to \{0, 1\}$ *for* $\mathbf{R}_i$ *is the delta function defined by*

$$\delta^{\mathcal{R}_i}(p) = \begin{cases} 1 \text{ if } p \text{ is preseved by } \mathbf{R}_i, \\ 0 \text{ else} \end{cases} \tag{3.26}$$

When we apply a projection $\mathcal{R}_i\mathbf{x}_i$, some of the entries of $\mathbf{x}_i$ are preserved, and we sometimes want to refer to these by their index. An entry in the pth position of $\mathbf{x}_i$ may get projected to the qth position in $\mathcal{R}_i\mathbf{x}_i$. We want a map $m_i : I_P \mapsto \{1, 2, \dots, Q\}$ that identifies the projected index for every preserved feature, the set of which we denote by I_P. The map has a dependence on i because the mask is different for each datapoint. For example, suppose we project

$$[y_1, y_2, y_3, y_4] \to [y_2, y_4].$$

Then $I_p = \{2, 4\}$ and $m_i(2) = 1, m_i(4) = 2$, since y_2 is now the first entry in the projection, and y_4 is the second. This is invertible, so that $m_i^{-1}(1) = 2$ and $m_i^{-1}(2) = 4$.

Definition 4 (Mask index)**.** *Let* $\mathbf{R}_i$ *be a sampling projection. The* ***mask index*** $m_i : I_P \mapsto \{1, 2, \dots, Q\}$ *is the function that gives the index in* $\mathbb{R}^Q$ *corresponding to each index in the set* I_P *indexing the entries preserved by* $\mathbb{R}_i$*.*

Derivatives of Sparsified Precision With Respect to Dense Covariance

We now consider derivatives of both subsampled and sparsified covariance ($\mathbf{S}$) and precision ($\mathbf{\Lambda}$) matrices. To find the maximum likelihood estimators we will need to compute quantities such as

$$\frac{\partial}{\partial \mathbf{S}(p_1, p_2)} \left({\mathbf{R}_i}^T \mathbf{S} \mathbf{R}_i \right)^{-1}.$$

To begin we find the entrywise derivative of the subsampled covariance, which we will need in our subsequent analyses.

Remark 5 (Entrywise Derivative of Subsampled Covariance)**.** *Let* $\mathbf{S} \in \mathbb{R}^{P \times P}$ *be a symmetric positive semi-definite matrix (such as a covariance matrix) and* $\mathbf{R}_i$ *be a sampling projection. Then*

$$\frac{\partial}{\partial \mathbf{S}(p_1,p_2)}\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i = \mathbf{E}^{\mathcal{R}_i}_{p_1p_2} \tag{3.27}$$

where the elementary matrix $\mathbf{E}^{\mathcal{R}_i}_{p_1p_2} \in \mathbb{R}^{Q\times Q}$ *is defined entrywise by*

$$\left[\mathbf{E}^{\mathcal{R}_i}_{p_1p_2}\right](q_1,q_2) = \begin{cases} 1 \text{ if } \delta^{\mathcal{R}_i}(p_1) = \delta^{\mathcal{R}_i}(p_2) = 1 \text{ and } q_1 = m_i(p_1), q_2 = m_i(p_2) \\ 0 \text{ else} \end{cases} \tag{3.28}$$

where $m_i(p)$ *is the mask index in definition 4.*

This is pretty ugly, but conceptually simple. $\mathbf{E}^{\mathcal{R}_i}_{p_1p_2}$ has up to one non-zero entry. If $\mathbf{R}_i$ doesn't preserve indices p_1 and p_2, then $\mathbf{E}^{\mathcal{R}_i}_{p_1p_2} = \mathbf{0}$. If $\mathbf{R}_i$ preserves both p_1 and p_2, then $\mathbf{E}^{\mathcal{R}_i}_{p_1p_2}$ has a single 1 in it, in the row corresponding to the masked index of p_1 and column corresponding to the masked index of p_2.

We will use the right action of $\mathbf{E}^{\mathcal{R}_i}_{p_1p_2}$. For $\mathbf{A} \in \mathbf{R}^{Q\times Q}$, then (when $\mathbf{E}^{\mathcal{R}_i}_{p_1p_2}$ is not zero):

$$\mathbf{A}\mathbf{E}^{\mathcal{R}_i}_{p_1p_2} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \dots & \mathbf{A}(:,m_i(p_1)) & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix}$$

where the non-zero column has index $m_i(p_2)$. Again the concept is simpler than the notation: $\mathbf{E}^{\mathcal{R}_i}_{p_1p_2}$ takes a column of $\mathbf{A}$ and puts it as a column in an otherwise-zero matrix. The column preserved from $\mathbf{A}$ corresponds to the masked index of p_1, and the column of the output into which it goes corresponds to the masked index of p_2.

Remark 6 (Derivative of $\mathbf{A}^{-1}$)**.** *For invertible* $\mathbf{A}(t) \in \mathbb{R}^{P\times P}$*,*

$$\frac{d}{dt}\mathbf{A}^{-1} = -\mathbf{A}^{-1}\left(\frac{d\mathbf{A}}{dt}\right)\mathbf{A}^{-1} \tag{3.29}$$

In particular, when $t = \mathbf{A}(p_1,p_2)$*, we have that*

$$\left[\frac{d}{d\mathbf{A}(p_1,p_2)}\mathbf{A}^{-1}\right](m,n) = -\left[\mathbf{A}^{-1}\right](m,p_1)\left[\mathbf{A}^{-1}\right](p_2,n) \tag{3.30}$$

Our next lemma derives the analog of equation (3.30) in the case of the precision matrix under sparsification.

Lemma 7 (Entrywise Derivative of Sparsified Precision)**.** *Let* $\mathbf{S} \in \mathbb{R}^{P\times P}$ *be a covariance matrix,* $\mathbf{R}_i$ *be a subsampling matrix, and* $\mathbf{\Lambda}^{\mathcal{R}_i}$ *be the* ***sparsified precision****:*

$$\mathbf{\Lambda}_k^{\mathcal{R}_i} = \mathbf{R}_i(\mathbf{R}_i{}^T\mathbf{S}_k\mathbf{R}_i)^{-1}\mathbf{R}_i{}^T \in \mathbb{R}^{P\times P} \tag{3.31}$$

Then,

$$\left[\frac{d\mathbf{\Lambda}^{\mathcal{R}_i}}{d\mathbf{S}(p_1,p_2)}\right](n_1,n_2) = -\left[\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i\right]^{-1}(m_i^{-1}(n_1),m_i(p_1))\left[\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i\right]^{-1}(m_i(p_2),m_i^{-1}(n_2)). \tag{3.32}$$

Proof. Applying remarks (6) and (5),

$$\frac{d}{d\mathbf{S}(p_1,p_2)}(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-1} = -(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-1}\mathbf{E}^{\mathcal{R}_i}_{p_1p_2}(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-1} \tag{3.33}$$

Now, $\mathbf{E}^{\mathcal{R}_i}_{p_1p_2}$ is idempotent, so we can write the above as

$$\begin{aligned}
\frac{d}{d\mathbf{S}(p_1,p_2)}(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-1} &= -(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-1}\mathbf{E}^{\mathcal{R}_i}_{p_1p_2}\mathbf{E}^{\mathcal{R}_i}_{p_1p_2}(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-1} && (3.34)\\
&= -(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-1}\mathbf{E}^{\mathcal{R}_i}_{p_1p_2}\left[(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-T}(\mathbf{E}^{\mathcal{R}_i}_{p_1p_2})^T\right]^T && (3.35)\\
&= -(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-1}\mathbf{E}^{\mathcal{R}_i}_{p_1p_2}\left[(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-T}\mathbf{E}^{\mathcal{R}_i}_{p_2p_1}\right]^T && (3.36)
\end{aligned}$$

As discussed in the comments following remark (5), the right action of $\mathbf{E}^{\mathcal{R}_i}_{p_1p_2}$ is to take the $m_i(p_1)$ column of $(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-1}$ and put it in the $m_i(p_2)$ column of a zero matrix. In the case of the second first consider the argument to the transpose

$$(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-T}\mathbf{E}^{\mathcal{R}_i}_{p_2p_1}$$

Here $\mathbf{E}^{\mathcal{R}_i}_{p_2p_1}$ takes the $m_i(p_2)$ column of $(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-T}$, which is to say the $m_i(p_2)$ row of $(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-1}$, and puts it into the $m_i(p_1)$ column of a zero matrix. Then

$$\left[(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-T}\mathbf{E}^{\mathcal{R}_i}_{p_2p_1}\right]^T$$

is a zero matrix with the $m_i(p_2)$ row of $(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-1}$ in its $m_i(p_1)$ row. Thus the product in equation (3.36) is equivalent to the outer product of the $m_i(p_1)$ column of $(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-1}$ and the $m_i(p_2)$ row of $(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-1}$:

$$\frac{d}{d\mathbf{S}(p_1,p_2)}(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-1} = -(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-1}(:,m_i(p_1))\circ(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-1}(m_i(p_2),:) \tag{3.37}$$

Thus the m, nth entry

$$\left[\frac{d}{d\mathbf{S}(p_1,p_2)}(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-1}\right](n_1,n_2) = -\left[(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-1}\right](n_1,m_i(p_1))\left[(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-1}\right](m_i(p_2),n_2) \tag{3.38}$$

For clarity define

$$\begin{aligned}\mathbf{L}_{p_1,p_2} &= \frac{d}{d\mathbf{S}(p_1,p_2)}(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-1} &(3.39)\\ &= -(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-1}(:,m_i(p_1))\circ(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)^{-1}(m_i(p_2),:) &(3.40)\end{aligned}$$

Then,

$$\begin{aligned}\left[\mathbf{R}_i{}^T\mathbf{L}_{p_1,p_2}\mathbf{R}_i\right](n_1,n_2) &= -\sum_{l,k}^{Q}\mathbf{R}_i{}^T(n_1,l)\mathbf{L}_{p_1,p_2}(l,k)\mathbf{R}_i(k,n_2) &(3.41)\\ &= \mathbf{L}_{p_1,p_2}(m_i^{-1}(n_1),m_i^{-1}(n_2)) &(3.42)\end{aligned}$$

because of the stucture of $\mathbf{R}_i$ - the n_1th row of $\mathbf{R}_i{}^T$ has exactly one non-zero entry, a 1 in the $m_i^{-1}(n_1)$ row indicating which dense feature is saved in the n_1th compressed feature and so both sums collapse. The desired result then follows. □

Remark 8 (Jacobi's formula)**.** *Let* $\mathbf{A}(t) \in \mathbb{R}^{P\times P}$ *have (entry-wise) dependence on* $t \in \mathbb{R}$*. Then*

$$\frac{d}{dt}\det\mathbf{A} = tr\left[adj(\mathbf{A})\frac{d\mathbf{A}}{dt}\right]. \tag{3.43}$$

where $adj(\mathbf{A})$ *is the* ***adjugate*** *of* $\mathbf{A}$*.*

When $\mathbf{A}$ is invertible the adjugate is

$$\mathrm{adj}(\mathbf{A}) = \det(\mathbf{A})\mathbf{A}^{-1},$$

and so remark 8 gives that

$$\frac{d}{dt}\det\mathbf{A} = \mathrm{tr}\left[\det(\mathbf{A})\mathbf{A}^{-1}\frac{d\mathbf{A}}{dt}\right].$$

This can be used to compute $\frac{d}{d\mathbf{A}}\det\mathbf{A}$:

Remark 9 (Derivative of $\det\mathbf{A}$)**.** *The entrywise derivative of* $\det\mathbf{A}$ *is*

$$\frac{d}{A(p,q)}\det\mathbf{A} = adj(\mathbf{A})(q,p) \tag{3.44}$$

and so the matrix derivative is

$$\frac{d}{d\mathbf{A}} \det \mathbf{A} = adj(\mathbf{A})^T. \tag{3.45}$$

When $\mathbf{A}$ *is invertible,*

$$\frac{d}{d\mathbf{A}} \det \mathbf{A} = \frac{1}{\det \mathbf{A}} \mathbf{A}^{-T}. \tag{3.46}$$

In turn this can be used to find $\frac{d}{d\mathbf{A}} \log \det \mathbf{A}$

Remark 10 (Derivative of $\log \det \mathbf{A}$)**.** *When* $\mathbf{A}$ *is invertible,*

$$\frac{d}{d\mathbf{A}} \log \det \mathbf{A} = \mathbf{A}^{-T} \tag{3.47}$$

When applied to a covariance matrix $\mathbf{S}$, which is symmetric, this gives

$$\frac{d}{d\mathbf{S}} \log \det \mathbf{S} = \mathbf{\Lambda} \tag{3.48}$$

where $\mathbf{\Lambda} = \mathbf{S}^{-1}$ is the **precision matrix**. Our goal now is to find the equivalent of this under sparsification.

Lemma 11 (Derivative of $\log \det$ under sparsification.)**.** *Let* $\mathbf{S} \in \mathbb{R}^{P \times P}$ *be a covariance matrix and* $\mathbf{R}_i \in \mathbb{R}^{P \times Q}$ *be a sampling projection. Then*

$$\frac{d}{d\mathbf{S}} \log \det \left({\mathbf{R}_i}^T \mathbf{S} \mathbf{R}_i \right) = \mathbf{\Lambda}^{\mathcal{R}_i} \tag{3.49}$$

where $\mathbf{\Lambda}^{\mathcal{R}_i}$ *is defined in equation (3.69).*

Proof. By the chain rule for matrices,

$$\frac{d}{d\mathbf{S}} \log \det \left({\mathbf{R}_i}^T \mathbf{S} \mathbf{R}_i \right) = \frac{1}{\det \left({\mathbf{R}_i}^T \mathbf{S} \mathbf{R}_i \right)} \frac{d}{d\mathbf{S}} \det \left({\mathbf{R}_i}^T \mathbf{S} \mathbf{R}_i \right). \tag{3.50}$$

Next, apply Jacobi's formula from remark (8) to find the element-wise derivative:

$$\begin{aligned} \frac{d}{d\mathbf{S}(p_1, p_2)} \det \left({\mathbf{R}_i}^T \mathbf{S} \mathbf{R}_i \right) &= \operatorname{tr} \left[\operatorname{adj}({\mathbf{R}_i}^T \mathbf{S} \mathbf{R}_i) \frac{d}{d\mathbf{S}(p_1, p_2)} ({\mathbf{R}_i}^T \mathbf{S} \mathbf{R}_i) \right] & (3.51) \\ &= \operatorname{tr} \left[\det({\mathbf{R}_i}^T \mathbf{S} \mathbf{R}_i) ({\mathbf{R}_i}^T \mathbf{S} \mathbf{R}_i)^{-1} \frac{d}{d\mathbf{S}(p_1, p_2)} ({\mathbf{R}_i}^T \mathbf{S} \mathbf{R}_i) \right] & (3.52) \end{aligned}$$

assuming both p_1 and p_2 are preserved by the mask (otherwise the previous derivative is 0). Using the notation in remark (5), then

$$\frac{d}{d\mathbf{S}(p_1,p_2)}(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i) = \mathbf{E}^{\mathcal{R}_i}_{p_1p_2},$$

and following the same logic thereafter,

$$\left(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i\right)\frac{d}{d\mathbf{S}(p_1,p_2)}(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i) = \left[\begin{array}{ccccccccc} \mathbf{0} & \mathbf{0} & \dots & \left[\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i\right](:,m_i(p_1)) & \mathbf{0} & \dots & \mathbf{0}. \end{array}\right]$$

The trace thus picks out one entry from this non-zero column; namely,

$$\mathrm{tr}\left[\frac{d}{d\mathbf{S}(p_1,p_2)}(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)\right] = \left[\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i\right]^{-1}(m_i(p_2),m_i(p_1)). \tag{3.53}$$

This is the analog of equation (3.45). We can replace the masked indexing by projecting the matrix back into P-space:

$$\left[\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i\right]^{-1}(m_i(p_2),m_i(p_1)) = \left[\mathbf{R}_i\left(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i\right)^{-1}\mathbf{R}_i{}^T\right](p_2,p_1). \tag{3.54}$$

This is valid whether p_1 and p_2 are preserved by the mask or not, since it correctly evaluates to 0 in the latter case. Susbtituting this into equation (3.52) we obtain

$$\frac{d}{d\mathbf{S}(p_1,p_2)}\det\left(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i\right) = \det(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)\left[\mathbf{R}_i\left(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i\right)^{-1}\mathbf{R}_i{}^T\right](p_2,p_1). \tag{3.55}$$

By definition,

$$\left[\mathbf{R}_i\left(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i\right)^{-1}\mathbf{R}_i{}^T\right] = \mathbf{\Lambda}^{\mathcal{R}_i}$$

and since this is symmetric (by virtue of $\mathbf{S}$ and $\mathbf{R}_i$ both being so) we can swap the p_1, p_2 indices in equation (3.55):

$$\frac{d}{d\mathbf{S}(p_1,p_2)}\det\left(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i\right) = \det(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)\left[\mathbf{\Lambda}^{\mathcal{R}_i}\right](p_1,p_2). \tag{3.56}$$

Thus

$$\frac{d}{d\mathbf{S}}\det\left(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i\right) = \det(\mathbf{R}_i{}^T\mathbf{S}\mathbf{R}_i)\mathbf{\Lambda}^{\mathcal{R}_i}. \tag{3.57}$$

Finally, substituting this back into equation (3.50), the determinants cancel and we obtain the desired result. □

Next we compute the derivative of a particular trace.

Remark 12 ("Trace trick")**.** *For* $\mathbf{A} \in \mathbb{R}^{P\times P}$ *and column vector* $\mathbf{a} \in \mathbb{R}^P$*, then*

$$\mathbf{a}^T\mathbf{A}\mathbf{a} = tr\left[\mathbf{a}\mathbf{a}^T\mathbf{A}\right]. \tag{3.58}$$

The quantity $\mathbf{a}^T\mathbf{A}\mathbf{a}$ is a scalar, so treating it as a 1×1 matrix, then $\mathbf{a}^T\mathbf{A}\mathbf{a} = \text{tr}\left[\mathbf{a}^T\mathbf{A}\mathbf{a}\right]$. The trace is cyclic so we can cycle the order of the matrix products to obtain the result. This reformulation helps compute derivatives of $\mathbf{a}^T\mathbf{A}\mathbf{a}$.

Remark 13 (Derivative of $\text{tr}\left(\mathbf{M}\mathbf{A}\right)$)**.** *For* $\mathbf{A}, \mathbf{M} \in \mathbb{R}^{P\times P}$*,*

$$\frac{d}{d\mathbf{A}^{-1}} tr(\mathbf{M}\mathbf{A}) = -\left[\mathbf{A}^{-1}\mathbf{M}\mathbf{A}^{-1}\right]^T. \tag{3.59}$$

The sparsified equivalent is stated in Lemma (14):

Lemma 14 (Derivative of $\text{tr}\left(\mathbf{M}\mathbf{\Lambda}^{\mathcal{R}_i}\right)$)**.** *Let* $\mathbf{S}$ *be a covariance matrix,* $\mathbf{R}_i$ *a subsampling projection,* $\mathbf{\Lambda}^{\mathcal{R}_i}$ *be the sparsified precision as defined in equation (3.69), and* $\mathbf{M} \in \mathbb{R}^{P\times P}$ *be symmetric. Then*

$$\frac{d}{d\mathbf{S}} tr\left[\mathbf{M}\mathbf{\Lambda}^{\mathcal{R}_i}\right] = -\mathbf{\Lambda}^{\mathcal{R}_i}\mathbf{M}\mathbf{\Lambda}^{\mathcal{R}_i} \tag{3.60}$$

Proof. We can write the trace in component form as

$$\text{tr}\left[\mathbf{M}\mathbf{\Lambda}^{\mathcal{R}_i}\right] = \sum_{m,n} \mathbf{M}(m,n)\mathbf{\Lambda}^{\mathcal{R}_i}(n,m) \tag{3.61}$$

Then,

$$\begin{aligned} \frac{d\text{tr}\left[\mathbf{M}\mathbf{\Lambda}^{\mathcal{R}_i}\right]}{d\mathbf{S}(p_1,p_2)} &= \sum_{m,n} \mathbf{M}(m,n)\left[\frac{d\mathbf{\Lambda}^{\mathcal{R}_i}}{d\mathbf{S}(p_1,p_2)}\right](n,m) && (3.62)\\ &= \sum_{m,n} -\mathbf{M}(m,n)\mathbf{\Lambda}^{\mathcal{R}_i}(n,p_1)\mathbf{\Lambda}^{\mathcal{R}_i}(p_2,m) && (3.63) \end{aligned}$$

by Lemma (7). We recognize that equation (3.63) is a matrix product:

$$\begin{aligned} \sum_{m,n} -\mathbf{M}(m,n)\mathbf{\Lambda}^{\mathcal{R}_i}(n,p_1)\mathbf{\Lambda}^{\mathcal{R}_i}(p_2,m) &= -\left[\mathbf{\Lambda}^{\mathcal{R}_i}\mathbf{M}\mathbf{\Lambda}^{\mathcal{R}_i}\right](p_2,p_1) && (3.64)\\ &= -\left[\mathbf{\Lambda}^{\mathcal{R}_i}\mathbf{M}\mathbf{\Lambda}^{\mathcal{R}_i}\right]^T(p_1,p_2) && (3.65) \end{aligned}$$

which is the equivalent of equation (3.59) in remark (13). Since $\mathbf{\Lambda}^{\mathcal{R}_i}$ and $\mathbf{M}$ are symmetric, the composition in equation (3.65) is also, and the desired result follows. □

This concludes the necessary expressions for the sparsified precision derivatives. Next we obtain the requisite expression for derivatives with respect to the means. It turns out that the standard case suffices, and we need not compute any sparsified analogs.

Remark 15. *For* $\mathbf{A} \in \mathbb{R}^{P\times P}$ *independent of column vector* $\mathbf{a} \in \mathbb{R}^P$*,*

$$\frac{d}{d\mathbf{a}}\left(\mathbf{a}^T\mathbf{A}\mathbf{a}\right) = \left(\mathbf{A}^T + \mathbf{A}\right)\mathbf{a}, \tag{3.66}$$

and by applying the chain rule to the special case,

$$\frac{d}{d\boldsymbol{\mu}}\left[(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{A}(\mathbf{x}-\boldsymbol{\mu})\right] = -(\mathbf{A}^T + \mathbf{A})(\mathbf{x}-\boldsymbol{\mu}). \tag{3.67}$$

3.4.2 EM for Sparsified Gaussian Mixtures

We now present our main result, the EM algorithm for sparsified Gaussian mixtures; i.e., the equivalents to the responsibility in equation (3.20) and the parameter MLE's in equations (3.23-3.25) under sparsification.

The sparsified analog of the squared Mahalanobis distance in equation (3.7) is

$$D^{\mathcal{R}}_{\boldsymbol{\theta}_k}(\mathbf{x}_i) = \left(\mathbf{x}_i - \boldsymbol{\mu}_k\right)^T \boldsymbol{\Lambda}_k^{\mathcal{R}_i}\left(\mathbf{x}_i - \boldsymbol{\mu}_k\right) \tag{3.68}$$

where

$$\boldsymbol{\Lambda}_k^{\mathcal{R}_i} = \mathbf{R}_i(\mathbf{R}_i{}^T\mathbf{S}_k\mathbf{R}_i)^{-1}\mathbf{R}_i{}^T \in \mathbb{R}^{P\times P} \tag{3.69}$$

is the sparsified analog[2] of the precision matrix $\boldsymbol{\Lambda}_k = \mathbf{S}_k^{-1}$. The sparsified Gaussian density is:

$$p^{\mathcal{R}}_k(\mathbf{x}_i \mid \boldsymbol{\theta}_k) = \frac{1}{2\pi^{Q/2}}\frac{1}{|\mathbf{R}_i{}^T\mathbf{S}_k\mathbf{R}_i|^{1/2}}\exp\left(-\frac{1}{2}D^{\mathcal{R}}_{\boldsymbol{\theta}}(\mathbf{x}_i)\right). \tag{3.70}$$

This can be taken to be a Q-dimensional Gaussian with mean $\mathbf{R}_i\boldsymbol{\mu}_k$ and covariance $\mathbf{R}_i{}^T\mathbf{S}_k\mathbf{R}_i$ evaluated at $\mathbf{R}_i\mathbf{x}_i$. Both $p^{\mathcal{R}}$ and $D^{\mathcal{R}}_{\boldsymbol{\theta}_k}(\mathbf{x}_i)$ are unbiased estimators of their dense counterparts when scaled by P/Q (see Figure 3.1).

[2] We note that $\boldsymbol{\Lambda}_k^{\mathcal{R}_i}$ is not equivalent to $\mathbf{R}_i\mathbf{R}_i{}^T\boldsymbol{\Lambda}_k\mathbf{R}_i\mathbf{R}_i{}^T$; i.e., the sparsified embedding of the precision matrix $\boldsymbol{\Lambda}_k$

The E-step is to compute the responsibility as given in equation (3.20). Under sparsification, the responsibility becomes

$$r_{ik}^{\mathcal{R}} = \frac{\pi_k p_k^{\mathcal{R}}(\mathbf{x}_i \mid \boldsymbol{\theta}^{t-1})}{\sum_{j=1}^{K} \pi_j p_j^{\mathcal{R}}\big(\mathbf{x}_i \mid \boldsymbol{\theta}^{t-1}\big)} \tag{3.71}$$

and hence the sparsified auxiliary function Q in equation (3.21) is:

$$Q^{\mathcal{R}}(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1}) = \sum_{i,k} r_{ik}^{\mathcal{R}} \log\left[\pi_k p_k^{\mathcal{R}}(\mathbf{x}_i \mid \boldsymbol{\theta})\right]. \tag{3.72}$$

Theorem 16 (Maximum Likelihood Estimators for Sparsified Gaussian Mixtures)**.** *The maximum likelihood estimator for π_k with respect to $Q^{\mathcal{R}}$ is*

$$\widehat{\pi}_k^{\mathcal{R}} = \frac{\sum_i r_{ik}^{\mathcal{R}}}{N}. \tag{3.73}$$

The maximum likelihood estimators for $\boldsymbol{\mu}_k$ and $\mathbf{S}_k$ are the solutions to the system

$$\boldsymbol{\mu}_k^{\mathcal{R}} = \left(\sum_i r_{ik}^{\mathcal{R}} \boldsymbol{\Lambda}_k^{\mathcal{R}_i}\right)^{\dagger} \sum_i r_{ik}^{\mathcal{R}} \boldsymbol{\Lambda}_k^{\mathcal{R}_i} \mathbf{x}_i \tag{3.74}$$

$$\sum_i r_{ik}^{\mathcal{R}} \boldsymbol{\Lambda}_k^{\mathcal{R}_i} = \sum_i r_{ik}^{\mathcal{R}} \boldsymbol{\Lambda}_k^{\mathcal{R}_i} \mathbf{M}_{ik} \boldsymbol{\Lambda}_k^{\mathcal{R}_i} \tag{3.75}$$

where

$$\mathbf{M}_{ik} = \big(\mathbf{x}_i - \boldsymbol{\mu}_k\big)\big(\mathbf{x}_i - \boldsymbol{\mu}_k\big)^T. \tag{3.76}$$

is the scatter matrix.

Proof. The component of $Q^{\mathcal{R}}$ with π_k-dependence is

$$\ell^{\mathcal{R}}(\pi_k) = \sum_{i,k} r_{ik}^{\mathcal{R}} \log \pi_k$$

from which the MLE in equation (3.73) can be derived by setting $\partial \ell^{\mathcal{R}} / \partial \pi_k = 0$ for each k simultanously and solving the resulting system. The components of $Q^{\mathcal{R}}$ with $\boldsymbol{\mu}_k$ and $\mathbf{S}_k$ dependence are

$$\ell^{\mathcal{R}}(\boldsymbol{\mu}_k, \mathbf{S}_k) = \sum_i r_{ik}^{\mathcal{R}} \left(\log |\mathbf{R}_i{}^T \mathbf{S}_k \mathbf{R}_i| + D_{\boldsymbol{\theta}_k}^{\mathcal{R}}(\mathbf{x}_i)\right). \tag{3.77}$$

To find $\partial \ell^{\mathcal{R}} / \partial \boldsymbol{\mu}_k$, we observe that

$$\frac{\partial}{\partial \boldsymbol{\mu}_k} D_{\boldsymbol{\theta}_k}^{\mathcal{R}}(\mathbf{x}_i) = -2\boldsymbol{\Lambda}_k^{\mathcal{R}_i}\big(\mathbf{x}_i - \boldsymbol{\mu}_k\big)$$

by remark (15) and the symmetry of $\mathbf{\Lambda}_k^{\mathcal{R}_i}$. Equation (3.74) then follows by setting $\partial \ell^{\mathcal{R}} / \partial \boldsymbol{\mu}_k = 0$ and rearranging.

We now find $\partial \ell^{\mathcal{R}} / \partial \mathbf{S}_k$. For the first term in the summand of equation (3.77), we have that

$$\frac{\partial}{\partial \mathbf{S}_k} \log |{\mathbf{R}_i}^T \mathbf{S}_k \mathbf{R}_i| = \mathbf{\Lambda}_k^{\mathcal{R}_i} \tag{3.78}$$

by Lemma (11). For the second term, we apply the "trace trick" from Remark (12):

$$D_{\boldsymbol{\theta}_k}^{\mathcal{R}}(\mathbf{x}_i) = \operatorname{tr}\left[\mathbf{M}_{ik} \mathbf{\Lambda}_k^{\mathcal{R}_i}\right] \tag{3.79}$$

to find

$$\frac{\partial}{\partial \mathbf{S}_k} D_{\boldsymbol{\theta}_k}^{\mathcal{R}}(\mathbf{x}_i) = -\mathbf{\Lambda}_k^{\mathcal{R}_i} \mathbf{M}_{ik} \mathbf{\Lambda}_k^{\mathcal{R}_i} \tag{3.80}$$

by Lemma (14). Setting $\partial \ell / \partial \mathbf{S}_k = 0$ from equation (3.77) and using equations (3.78) and (3.80) we obtain equation (3.75). □

Evaluating these MLEs does not require access to the full $\mathbf{x}_i$, as in each case such terms are sparsified by the action of $\mathbf{\Lambda}_k^{\mathcal{R}_i}$. In the case of no sparsification; i.e., $\mathbf{R}_i = I$ for all i, we recover the standard MLEs in equations (3.23 - 3.25). Equation (3.73) has only $\pi_k^{\mathcal{R}}$ dependence, and hence gives the MLE for this parameter. Equation (3.74) gives the MLE for $\boldsymbol{\mu}_k^{\mathcal{R}}$ in terms of the $\mathbf{\Lambda}_k^{\mathcal{R}_i}$. In the standard case, the $\mathbf{\Lambda}_k$ terms cancel and we obtain the MLE for $\boldsymbol{\mu}_k$, which is then used in place of $\boldsymbol{\mu}_k$ to find the MLE for $\mathbf{S}_k$; however, in the sparsified case we do not observe this cancelation, and hence must solve equations (3.74) and (3.75) simultaneously. This can be done, for example, in an EM-type iterative fashion, but such a procedure further requires the evaluation of $\mathbf{\Lambda}_k^{\mathcal{R}_i}$, involving a $Q \times Q$ inverse, of which there are KN per iteration. These issues can be circumvented by using diagonal or spherical covariances. We give the MLEs for the diagonal case, $\mathbf{S}_k = \operatorname{diag}(\mathbf{s}_k)$ where $\mathbf{s}_k \in \mathbb{R}^P$.

Corollary 17 (MLEs for diagonal $\mathbf{S}_k$)**.** *When the $\mathbf{S}_k$ are diagonal, the system of equations (3.74*

- 3.75) yields the MLEs

$$\widehat{\boldsymbol{\mu}}_k = \left(\sum_i r_{ik}^{\mathcal{R}} \mathbf{P}_i\right)^{\dagger} \sum_i r_{ik}^{\mathcal{R}} \mathbf{P}_i \mathbf{x}_i \tag{3.81}$$

$$\widehat{\mathbf{s}}_k = diag\left[\left(\sum_i r_{ik}^{\mathcal{R}} \mathbf{P}_i\right)^{\dagger} \sum_i r_{ik}^{\mathcal{R}} \mathbf{P}_i \mathbf{M}_{ik} \mathbf{P}_i\right] \tag{3.82}$$

where $\mathbf{P}_i \in \mathbb{R}^{P \times P}$ *is the sparse projection matrix:*

$$\mathbf{P}_i = \mathbf{R}_i \mathbf{R}_i^T. \tag{3.83}$$

Proof. When $\mathbf{S}_k = \text{diag}(\mathbf{s}_k)$, then

$$\left(\mathbf{R}_i^T \mathbf{S}_k \mathbf{R}_i\right)^{-1} = \mathbf{R}_i^T \mathbf{S}_k^{-1} \mathbf{R}_i.$$

Since $\mathbf{R}_i$ is also diagonal, then we can commute $\text{diag}(\mathbf{s}_k)$ out of the sum in the first term of equation (3.74):

$$\left(\sum_i r_{ik}^{\mathcal{R}} \boldsymbol{\Lambda}_k^{\mathcal{R}_i}\right)^{\dagger} = \left(\sum_i \mathbf{R}_i^T \mathbf{R}_i\right)^{\dagger} \text{diag}(\mathbf{s}_k)$$

where we have also used that $\mathbf{R}_i^T \mathbf{R}_i$ is idempotent. The same argument applies to the second term in equation (3.74):

$$\sum_i r_{ik}^{\mathcal{R}} \boldsymbol{\Lambda}_k^{\mathcal{R}_i} \mathbf{x}_i = \text{diag}(\mathbf{s}_k)^{-1} \sum_i \mathbf{R}_i^T \mathbf{R}_i \mathbf{x}_i.$$

Combining these results gives the MLE $\widehat{\boldsymbol{\mu}}_k$ in equation (3.81). A similar argument applies for $\widehat{\mathbf{S}}_k$ in equation (3.82).

□

Corollary 18 (MLEs for spherical $\mathbf{S}_k$)**.** *When the* $\mathbf{S}_k$ *are spherical, the system of equations (3.74 - 3.75) yields the MLEs*

$$\widehat{\boldsymbol{\mu}}_k = \left(\sum_i r_{ik}^{\mathcal{R}} \mathbf{P}_i\right)^{\dagger} \sum_i r_{ik}^{\mathcal{R}} \mathbf{P}_i \mathbf{x}_i \tag{3.84}$$

$$\widehat{s}_k = \frac{1}{P} \sum_{p=1}^{P} \left[\left(\sum_i r_{ik}^{\mathcal{R}} \mathbf{P}_i\right)^{\dagger} \sum_i r_{ik}^{\mathcal{R}} \mathbf{P}_i \mathbf{M}_{ik} \mathbf{P}_i\right]_{pp} \tag{3.85}$$

where $\mathbf{P}_i$ *is as defined in equation (3.83).*

Proof. The MLE for $\boldsymbol{\mu}_k$ is independent of $\mathbf{S}_k$ and is hence the same as in the diagonal case. Since $\mathbf{S}_k = s_k I$ then, as in the diagonal case, we can factor this term out of both sides of equation 3.75 to obtain

$$\widehat{s}_k = \text{argmax}_{s_k} \|s_k \mathbf{I} - \mathbf{B}\| \tag{3.86}$$

where

$$\mathbf{B} = \left(\sum_i r_{ik}^{\mathcal{R}} \mathbf{P}_i \right)^{\dagger} \sum_i r_{ik}^{\mathcal{R}} \mathbf{P}_i \mathbf{M}_{ik} \mathbf{P}_i. \tag{3.87}$$

Choosing the Frobenius norm, the solution is

$$\widehat{s}_k = \frac{1}{P} \sum_{p=1}^{P} [\mathbf{B}]_{pp}\,; \tag{3.88}$$

i.e., the mean diagonal of $\mathbf{B}$. Note that this is equivalent to the mean of $\mathbf{s}_k$, the diagonal MLE in equation 3.82. □

The EM algorithm for SGMM is presented in Algorithm 3.2. The algorithm begins with sparsified, preconditioned data $\mathcal{X}$, the mask indices M, and initial model parameters θ_0. The E and M steps are repeated until convergence or until the maximum iterations has been reached, and returns the parameter estimates $\boldsymbol{\theta}^t$ after t iterations.

3.4.3 Computational Complexity

In the case of diagonal covariances (as well as in the simpler spherical case in which $\mathbf{S}_k = s_k I$), the responsibilities $r_{ik}^{\mathcal{R}}$ (E step) and the updates for $\widehat{\boldsymbol{\mu}}_k$ and $\widehat{\mathbf{S}}_k$ (M step) can be each be computed in $\mathcal{O}(KNQ)$ time (see Table (3.1)). Thus the EM algoritm has time complexity $\mathcal{O}(KNQ)$ per iteration, in contrast to the standard diagonal GMM's complexity of $\mathcal{O}(KNP)$ per iteration.

For concreteness we include complexity computations in Table 3.3 using characteristic parameter values given in Table 3.4.

Algorithm 3.2: EM for sparsified Gaussian mixture model.

Inputs : $\mathcal{R}X$, sparsified, preconditioned data.
$\mathbf{M}$, mask indices.
θ_0, initial parameter estimates.
$\Delta\theta_{tol}$, convergence tolerance.
t_{max}, maximum iterations.

Output: $\boldsymbol{\theta}^t$

initialize $\Delta\theta, t$;
while $(\Delta\theta > \Delta\theta_{tol})$ **and** $(t \leq t_{max})$ **do**
  `// E step`
  for $k = 1, 2, \ldots, K$ **do**
    for $n = 1, 2, \ldots N$ **do**
      compute responsibility $r_{nk}^{\mathcal{R}}$ using $\boldsymbol{\theta}^{t-1}$;
    end for
  end for
  `// M step`
  $\{\boldsymbol{\theta}^t, \boldsymbol{\pi}\} \leftarrow \text{argmax}_{\boldsymbol{\theta},\boldsymbol{\pi}} Q^{\mathcal{R}}(\theta, \theta^{t-1})$;
  `// Convergence checks`
  $\Delta\theta \leftarrow \left\|\boldsymbol{\theta}^t - \boldsymbol{\theta}^{t-1}\right\|$;
  $t \leftarrow t + 1$;
end while

Table 3.1: Computational complexity comparison between GMM and SGMM, for full and diagonal covariances. See Table (3.2) for parameter descriptions.

Step	GMM, full $\mathbf{S}$	SGMM, full $\mathbf{S}$	GMM, diag $\mathbf{S}$	SGMM, diag $\mathbf{S}$
E step	$\mathcal{O}(NKP^2 + KP^3)$	$\mathcal{O}(NKQ^3)$	$\mathcal{O}(NKP)$	$\mathcal{O}(NKQ)$
M step	$\mathcal{O}(NKP^2)$	$\mathcal{O}(NKQ^2)$	$\mathcal{O}(NKP)$	$\mathcal{O}(NKQ)$
Total	$\mathcal{O}(INKP^2 + IKP^3)$	$\mathcal{O}(INKQ^3)$	$\mathcal{O}(INKP)$	$\mathcal{O}(INKQ)$

Table 3.2: Parameter descriptions.

Parameter	**Description**
N	number of points
K	number of components
P	ambient data dimension
Q	compressed ambient data dimension
I	number of EM iterations

Table 3.3: Computational complexity of SGMM for characteristic parameter values.

Step	GMM, full **S**	SGMM, full **S**	GMM, diag **S**	SGMM, diag **S**
E step	10^6	10^4	10	.1
M step	10^6	100	10	.1
Total	10^7	10^5	100	1

Table 3.4: Characteristic parameter values for SGMM.

Parameter	**Description**
$N = 10^6$	number of points
$K = 5$	number of components
$P = 10^4$	ambient data dimension
$Q = 100$	compressed ambient data dimension
$I = 10$	number of EM iterations

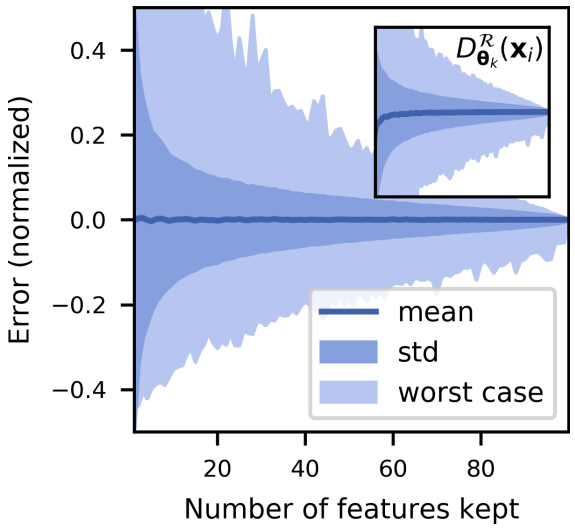


Figure 3.1: Error in $p_k^{\mathcal{R}}$ as a function of compression. 10000 $\mathbf{x}_i \sim \mathcal{N}(0, 1)$ in 100 dimensions per trial. Inset: error in $D^{\mathcal{R}}_{\boldsymbol{\theta}_k}(\mathbf{x}_i)$.

3.5 Simulations

3.5.1 Accuracy and Timing on MNIST

Figure 3.2 shows the accuracy of the SGMM classifier on the subset $\{0, 3, 9\}$ of the MNIST dataset as a function of the percentage of features preserved. SGMM recovers close to full GMM accuracy with only a small number of features in a fraction of the time. For instance, at the gray dot, SGMM with 3.82% of the features preserved (30 out of 784) achieves a mean accuracy of 0.86 (92% of the accuracy with all features) in 12.9% of the computation time excluding the preconditioning cost (which took 0.52 seconds) or 19.9% including the preconditioning. We further note that there is almost no variance in the accuracy over multiple trials; a consequence of our sampling scheme that we also observed in our sparsified k-means classifier [46].

3.5.2 Small Cluster Recovery

In a regime where clusters have very different sizes, both in the sense of variance and number of points, GMM (even with spherical covariance) can significantly outperform k-means. Figure 3.3 shows an example of this, where SGMM correctly identifies two small clusters from one large one on

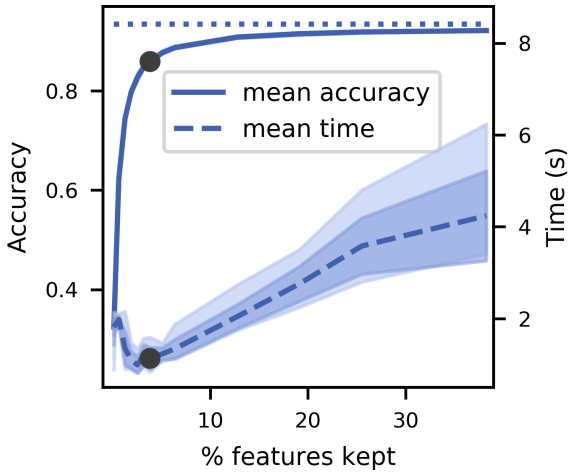


Figure 3.2: Accuracy and timing of diagonal SGMM on the subset {0,3,9} of MNIST ($N = 18003$) as a function of compression. 3 initializations per trial, 20 trials per compression. Shaded regions indicate standard deviation (dark) and extrema (light) taken over the trials.

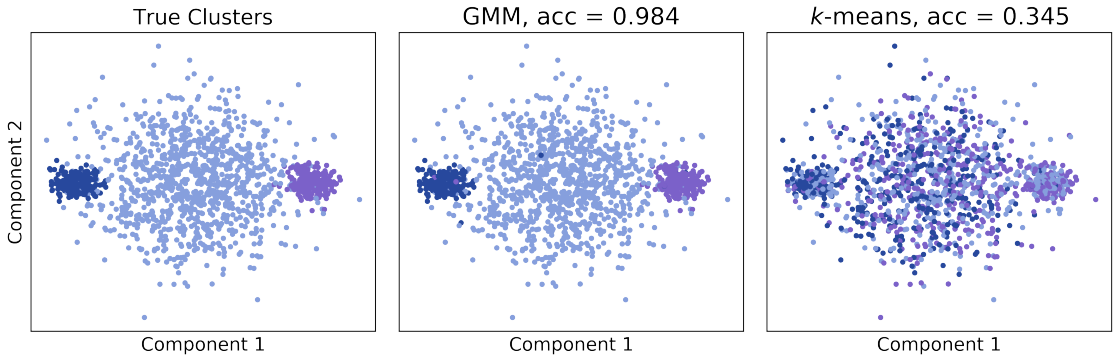


Figure 3.3: Small cluster recovery using spherical SGMM. See supporting text for details.

simulated data drawn from three Gaussians with $\|\mathbf{S}_{big}\|_2 \sim 10\,\|\mathbf{S}_{small}\|_2$ and $N_{big} = 5N_{small} = 1000$. In this simulation $P = 100$ and $Q = 10$.

3.6 Conclusions

3.6.1 Summary

The sparsified Gaussian mixture model is an efficient clustering algorithm that reduces the storage and computational cost of Gaussian mixtures while still being one-pass. After paying an upfront cost of $\mathcal{O}(NP \log P)$ to precondition the data, SGMM compresses N samples from P to Q dimensions, and with diagonal or spherical covariances, fits K clusters in $\mathcal{O}(KNQ)$ time per EM iteration.

3.6.2 Extensions and Future Work

We conclude with several potentially fruitful extensions for this work.

One-step bounds

Our main theoretical result is the derivation of maximum likelihood estimators under sparsification for the M step of the EM algorithm. The sparsification scheme we use here has been shown to satisfy several probabilistic bounds. It may be possible to extend some of the results to the SGMM algorithm. One such result is that, when applied to k-means, the sparsification scheme

satisfies a type of one-step update bound: the error between the update to a sample mean in one iteration of the sparsified algorithm and the same sample mean without sparsification is bounded probabilistically [46, Theorem 8]. Let $\bar{\boldsymbol{x}}_k$ denote the sample mean of the kth cluster. Then for all $t \geq 0$:

$$\frac{1}{\sqrt{P}} \|\boldsymbol{\mu}_k - \bar{\boldsymbol{x}}_k\| \leq t \left(1 + \frac{1}{\sqrt{P}\,\|\boldsymbol{\mu}\|_k}\right) \tag{3.89}$$

with probability greater than $1 - \delta$, where δ depends on the sample, P and Q, and t. It is likely that we could establish a similar bound for the M step parameter estimation in the EM algorithm. To do so we could apply other results in [46]; namely, (1) a theorem bounding

$$P\left(\|\sum_{i \in C_k} \mathbf{R}_t \mathbf{R}_i^T\| \leq t\right) \geq 1 - \delta \tag{3.90}$$

for all t with failure probability δ, and (2) theorems bounding the mean and covariance of the k-means clusters. This approach seems likely to be a straightforward but non-trivial extension of the existing results on k-means.

Alternative initialization strategies

The EM algorithm requires that the parameters be initialized. We currently use a sparsified variant of the well-known k-means++ initialization [79], in which a new initial mean is chosen randomly from the dataset with probability proportional to the squared distance of each point in the dataset from the current set of initial means. It may be possible to design an initialization strategy that is closer in spirit to Gaussian mixtures, say by setting the probability of selecting a point to instead be proportional to some Gaussian distribution. To do so we would need to devise a way to bootstrap covariances and weights in addition to the means.

Low-rank, full-dimensional covariances

We currently do not use the full covariances in our SGMM, instead restricting ourselves to the diagonal or spherical special cases. This is because it is computationally infeasible to compute N matrix inversions of the $Q \times Q$ submatrices of the covariance $\mathbf{S}_k$. As a result we lose some of the generality and flexibility of Gaussian mixtures. The idea of using low-rank approximations to

$\mathbf{S}_k$ is appealing; however, the quantities we need to compute in the updates,

$$\mathbf{\Lambda}_k^{\mathcal{R}_i} = \mathbf{R}_i \left(\mathbf{R}_i^T \mathbf{S}_k \mathbf{R}_i\right)^{-1} \mathbf{R}_i^T,$$

require inversions of distinct submatrices of $\mathbf{S}_k$. There are up to N such submatrices for each of K covariance matrices. If there is a way to connect the quantity $\mathbf{\Lambda}_k^{\mathcal{R}_i}$ with

$$\left(\mathbf{R}_i \mathbf{R}_i^T \mathbf{S}_k \mathbf{R}_i \mathbf{R}_i^T\right)^{-1}$$

then perhaps we could devise a way to permit low-rank dense covariances in SGMM.

Gaussian processes

A Gaussian process is a stochastic process such that every finite linear combination of its random variables is normally distributed [80]. Gaussian processes are widely used for regression and classification tasks in machine learning, where they offer a computationally tractable approach that is fully Bayesian in its reasoning. When the latent dimension is very high, however, Gaussian processes are susceptible to same issues as Gaussian mixture models. We are interested to see if the sparsification scheme we developed here, in particular the sparsified equivalents of distances and Gaussian probability distributions in equations (3.68 - 3.70), could be applied to Gaussian processes to create a faster and more memory-efficient classifier.

Chapter 4

Oracle Epiphany in Bayesian Networks

4.1 Introduction

In the previous chapter, we discussed a clustering algorithm designed for use on high-dimensional, streaming datasets. The goal of that method was to perform unsupervised learning on real-valued vector input $\{\mathbf{x}_i\} \subset \mathbb{R}^P$ efficiently and quickly by reducing the latent dimension of the data from P to Q with $Q \ll P$. In this chapter we focus on the related problem of needing to reduce the number of observations. Our problem setting will again be that of streaming data, and so it doesn't always make sense to draw the simple parallel that we wish to use M datapoints instead of $N \gg M$, but this description is accurate in spirit.

The work we present here is carried out with Respond software, a cyber security company building an artificial intelligence analyst to monitor network traffic with potentially many users (workstations at an office, for example) and identify cyber threats. For compliance with intellectual property concerns and because much of the data is inherently sensitive, there are parts of the work and the dataset that we do not discuss in detail. These omissions should not interfere with the clarity of exposition nor with the communication of our main findings: the goal of this chapter is to present a framework and an approach to solving a particular type of problem; and in this light the data we use to demonstrate our approach is incidental.

4.1.1 Problem Setting

The problem setting is the following. We observe a streaming flow of very high volume data, say on the of order one billion datapoints per day. Each individual datapoint $\mathbf{x}_i$ is not very large. In our specific application $\mathbf{x}_i$ consists of some short strings, a time-stamp, and maybe some vendor-specific telemetry about a particular kind of cyber threat. In general the approach we take is designed to handle multivariate categorical data. Additional structure, such as a natural language processing approach to string contents, agent-based or network-style frameworks, and time-series hierarchies, are each potentially relevant, but are not considered here. Thus, for our purposes, we assume

$$\mathbf{x} \sim \{C^{(1)}, C^{(2)}, \ldots, C^{(Q)}\} \tag{4.1}$$

where $C^{(q)}$ is a categorical random variable with K_q states. In general, we don't know anything a priori about the dependencies of the $C^{(q)}$; ultimately, however, we will use a Bayesian network, formalizing very specific assumptions. Our ultimate goal is to construct a parametric classifier Φ_θ to identify whether a particular observation constitutes a threat, which we formalize as a problem of binary classification: we seek to define Φ and find an estimate of θ such that

$$T = \Phi_\theta(\mathbf{x}), \tag{4.2}$$

with $T \in \{0, 1\}$, identifies whether $\mathbf{x}$ is benign ($T = 0$) or malicious ($T = 1$). We make no assumptions about the size of Q or $\{K_q\}$, but neither do we make any attempt for our method to scale well with either. This is to say that it's quite possible to run into trouble if either is too large. Our focus is instead to reduce how many datapoints we have to look at. Our particular domain of application, the identification of cyber threats, introduces a multitude of other practical and statistical constraints and considerations, none of which is necessarily unique to the domain at hand.

The biggest problem we face, in addition to the voluminous quantity of data, is that it is **very** expensive to obtain the true labels T_i. Like, really, truly, astronomically expensive, in every sense of the word. To begin with, the raw data we observe, call it $\mathbf{x}_i^{raw}$, is at the outset impossible

to label, probably comparably difficulty to looking at the array

$$\begin{bmatrix} 241 & 241 & 242 & \dots & 246 & 246 & 246 \\ 243 & 243 & 244 & \dots & 246 & 246 & 246 \\ 245 & 245 & 246 & \dots & 246 & 246 & 246 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 169 & 171 & 174 & \dots & 20 & 19 & 18 \\ 173 & 173 & 174 & \dots & 20 & 19 & 18 \\ 181 & 179 & 175 & \dots & 20 & 19 & 18 \end{bmatrix}$$

and recognizing it as the red channel of this picture of a cat:

Figure 4.1: A cat. Image by Dogbert420 (a suspicious name for an uploader of cat pictures). License: CC BY-SA 4.0. `https://en.wikipedia.org/wiki/Cat#/media/File:Close_up_of_a_black_domestic_cat.jpg`

We do not have the equivalent of image rendering software, and we must therefore engage in feature engineering before we know the labels T_i of any data. In practice, getting a label for a datapoint requires processing the raw datapoint $\mathbf{x}_i^{raw}$ into something a human cyber security expert recognizes as either benign activity ($T = 0$) or something scary enough to warrant a closer look ($T = 1$). For our purposes here, we encapsulate the entirety of this activity under the title of **querying the oracle**, terminology from the active learning literature [81].

The field of active learning concerns itself with the scenario in which the learning process itself is somehow used to decide which data gets labeled to be incorporated into some supervised

learning process [81]. Whence the name **active learning**: the personified learner actively chooses what she will learn from next. This approach is useful to us because we begin our process with no labeled data whatsoever, and it is excruciatingly difficult to get the labels. We therefore want to choose as carefully as we can which data get labeled.

Typically in active learning the concern is to choose a datapoint that will inform a region of parameter space about which we know little, or that will maximally improve the parameter estimates in some sense. In our case, however, we face the practical reality that not all queries are equivalently easy. We leverage recent work formalizing this concept into the framework of **oracle epiphany** [82]. The Oracle may not know the labels to a series of datapoints until suddenly something about the set triggers an epiphany, whereby she is instantaneously able to classify the entire set at once. The actual practical matter of inducing such epiphanies is subjective and touches on the vast domains of **knowledge engineering** [83] and **expert systems** [84].

We have so far described the related problems of **expensive Oracle queries** and **no labeled data**, which are all the more troublesome because of the vast volume of **streaming data** we will observe. The other major obstacle related to labeled data is that we observe **extreme class imbalance**: almost all traffic is benign with only a tiny minority constituting a real threat:

$$P(T_i = 1 \mid \mathbf{x}_i) \ll P(T_i = 0 \mid \mathbf{x}_i). \tag{4.3}$$

This only exacerbates the existing problems surrounding the acquisition of labeled data. These are the crucial issues we seek to address in the content presented here, but the system at hand has several other properties as well, which may consider in later work. These are:

- **Extreme Class Imbalance**: The vast majority of traffic is benign, incidents are very rare.
- **Mixed Datatype:** Raw data includes textual, temporal, categorical, and real number data types.
- **Time Series:** There is a clear temporal component to most of the data.

- **Heirarchical:** Data can be organized into several heirarchies, including users, sources, destinations, type of traffic, etc.
- **No Labeled Data:** We begin with no labeled data.
- **Expensive Oracle Queries:** Obtaining a label y for an event x is expensive.
- **Dependent Data:** Events are not independent; in particular, labeling an event in isolation is impossible.
- **Streaming Data:** The volume of data grows without bound and therefore not all of it can be kept. Data must be labeled in a streaming fashion.
- **Online Learning:** We need to be able to update the model parameters quickly, almost in real time, as the nature and type of threats we observe are constantly in flux.

Some of the issues discussed above require significant and unrelated efforts to address, and much of this work is presently ongoing.

4.1.2 Main Result and Chapter Overview

The **main result** we present in this chapter is a subset of the work described above. This is the application of oracle epiphany to a Bayesian network in order to simultaneously:

(1) reduce the number of datapoints from N to M with $M \ll N$

(2) query the Oracle for increased label coverage

(3) increase the relative number of the minority class to address class imbalance

Our results are preliminary and experimental. The work herein is as much a proof-of-concept as anything else. Luckily it seems to work OK, which is good since I hung both the completion of my PhD and my subsequent employment on this idea.

The chapter is organized as follows. First, in Section 4.2, we discuss alternative and complementary approaches to the **labeled data problem**. There are a lot of relevant ideas in the

literature, unfortunately for us most of them are not easily generalized beyond their specific domain of application and into ours. Then, we introduce Bayesian networks and discuss their applicability to our problem in Section 4.3. In particular we discuss learning algorithms to estimate model parameters θ and **sensitivity analysis**, a tool which we use to help guide our search for new label acquisition. In Section 4.4 we introduce active learning and oracle epiphany, and connect these concepts to the framework of Bayesian networks. In Section 4.5 we show the results of applying our approach to the problem of labeled data generation. Finally in Section 4.6 we discuss ongoing and future work.

4.2 Other Approaches To The Labeled Data Problem

We now discuss potential approaches to solving the problem of labeled data acquisition: one-shot learning, a Bayesian approach in computer vision to learn from very few training examples, the application of reinforcement learning to the task of classification. We consider these alternatives because each permits a form of classification in a scenario with little or no labeled data.

4.2.1 One-Shot Learning

One-shot learning is a classification task in which a category is learned from very few or even a single example [85]. The idea is to use information from known categories to learn new ones with much less training data by formulating the problem in a Bayesian framework. It originates in computer vision and most of its applications are in that field [86, 87, 85]. In the original formulation of the method [87, 85], the goal is to determine whether a given image $\mathcal{I}$ contains an example of a **foreground category**, denoted $\mathcal{O}_{fg}$, as opposed to the alternative of containing only generic **background clutter** $\mathcal{O}_{bg}$, given a set of training images for the foreground category, $\mathcal{I}_t$. This decision is made using the ratio

$$R = \frac{p\left(\mathcal{O}_{fg} \mid \mathcal{I}, \mathcal{I}_t\right)}{p\left(\mathcal{O}_{bg} \mid \mathcal{I}, \mathcal{I}_t\right)} \tag{4.4}$$

which by Bayes' Rule is equivalent to

$$R = \frac{p\left(\mathcal{I} \mid \mathcal{I}_t, \mathcal{O}_{fg}\right) p\left(\mathcal{O}_{fg}\right)}{p\left(\mathcal{I} \mid \mathcal{I}_t, \mathcal{O}_{bg}\right) p\left(\mathcal{O}_{bg}\right)}. \tag{4.5}$$

If R exceeds some specified threshold T then we conclude that I contains an instance of the foreground category. Choosing a parametric model for the background and foreground categories, with parameter vectors θ_{bg} and θ respectively, then R can be written

$$R \propto \frac{\int p\left(\mathcal{I} \mid \theta, \mathcal{O}_{fg}\right) p\left(\theta \mid \mathcal{I}_t, \mathcal{O}_{fg}\right) d\theta}{\int p\left(\mathcal{I} \mid \theta_{bg}, \mathcal{O}_{bg}\right) p\left(\theta_{bg} \mid \mathcal{I}_t, \mathcal{O}_{bg}\right) d\theta_{bg}}. \tag{4.6}$$

The learning phase consists of estimating $p(\theta \mid \mathcal{I}_t, \mathcal{O}_{fg})$; i.e., the posterior for the foreground category over θ given the training data for that categotry. In numerical experiments on image data in the original publication, the authors were able to achieve 75-90% accuracy given only 1-5 training examples.

The parametric model used in the original work is a Constellation model, which is specific to computer vision, for use with images represented as arrays of integers or floats. The framework presented above up through equation (4.6) is general, and hence given a different choice of parameterized model it could, in theory, apply to our domain of interest. The one-shot aspect of the method, however, seems to be intimately related to the Constellation model and image structure, and it is not evident how to preserve this property using a different model. There is at least one extension of one-shot learning to a different field: drug discovery in medicinal chemistry [88], and we therefore aim to consider adapting the method to our domain in the future.

4.2.2 Reinforcement Learning

Reinforcement learning (RL) is an independent domain of machine learning, alongside unsupervised learning and supervised learning, with the latter encompassing classification. Nevertheless there have been applications of reinforcement learning to the task of classification [89, 90, 91, 92]. The primary appeal of RL for our task is that the learning process typically takes place in the absence of labeled data. The standard framework for RL is a Markov decision process (MDP). An

MDP consists of the state S of the process, a finite set A of actions, a Markovian transition model P, and a reward function R (see for example [93] c.f. [89]). Denote by $\pi(s)$ the action the agent takes at state s, then the **value** of s under π is

$$V_\pi(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \mid s_0 = s\right] \tag{4.7}$$

where $\gamma \in [0, 1]$ is the **discount rate** for future rewards. V_π is the expected, total, discounted reward for the policy π and initial state s. The goal is then to find π^* that maximizes V_π, called an **optimal policy**.

Recently, reinforcement learning has been used to create AI with superhuman performance in the game Go, called AlphaGo Zero [94]. This framework has since been extended to a more general formulation, Alpha Zero, achieving the same results for the games of chess, Go, and Shogi [95]. In addition to superhuman performance, this program also convincingly beat all state-of-the-art AI game software (100-0 in the case of Go). The previous generation of AI game players were built using supervised or semi-supervised methods, incorporating enormous quantities of historical game data and domain knowledge. In contrast, RI approaches need only the rules of the game, learning from random play. Our system does not afford an obvious set of rules nor a clear reward function, and as such for the time being we will not consider reinforcement learning for our application.

4.3 Bayesian Networks

Bayesian networks are widely-used statistical models that represent dependencies between random variables using a directed acyclic graph. They were first introduced in 1985 [96], and two classical textbooks written shortly thereafter established their core theoretical foundation [97, 98]. A more recent textbook consolidates the main developments over the last three decades and presents the material in a modern perspective [99]. Bayesian networks were originally introduced to provide a tractable and intuitive formalization of human decision-making and reasoning about uncertainty [96].

Let $\mathcal{G}$ be a directed, acyclic graph with nodes representing categorical random variables

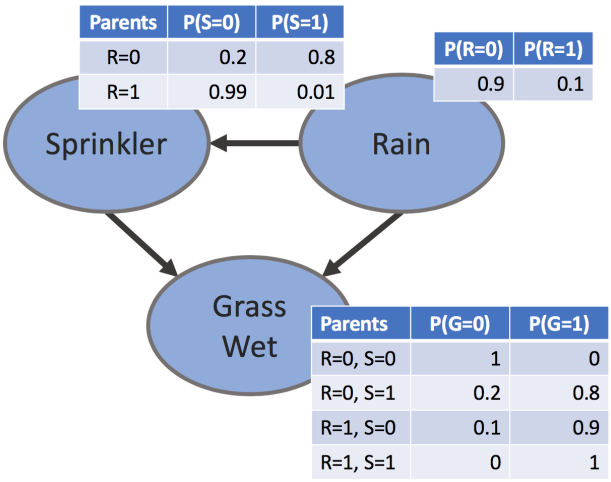


Figure 4.2: Example Bayesian network.

$\{X_1, X_2, \ldots, X_K\}$. A directed edge in G points from a **parent** to a **child**. If X is a parent of Y we write $X \to Y$. If $X_{i_1} \to X_{i_2} \to \ldots \to X_{i_m}$ then $\{X_{i_j}\}_{j=2}^{m}$ are **descendants** of X_{i_1}. Let $\mathbf{Pa}_i$ denote the parents of X_i and let $\mathbf{pa}_i$ enumerate all combinatoric states the parents can take. Note that we are indifferent here to the order in which they are listed. The parameters of the model θ are the conditional probabilities of each node given its parents:

$$\theta_{ijk} = P\left(X_i = x_i^k \mid \mathbf{Pa}_i = \mathbf{pa}_i^j\right). \tag{4.8}$$

In words, θ_{ijk} is the probability that X_i takes on state k given that the parents of X_i are in the jth configuration.

It is easiest to understand this by way of an example. Figure (4.2) shows a simple Bayesian network. This network models three random variables corresponding to (1) whether the grass is wet (G), (2) whether the sprinkler was on (S), and (3) whether it rained (R). R is a Bernoulli random variable with parameter $\alpha = 0.1$; i.e., the probability that it will rain is 0.1. The edge between R and S indicates a conditional dependency; the probability that the sprinkler was on depends on whether or not it has rained. For instance, $P(S = 1 \mid R = 0) = 0.8$; i.e., the probability that the sprinkler will go on given that it has not rained is 0.8, This is reflected in the (1,2) entry of of the table associated with S. The node G has two parents, and hence a larger table - we must

now account for all four possible combinations of parent states.

The parameters θ_{ijk} are stored in tables in this example; in particular, θ_{ijk} is the (j,k) entry of the ith table. If there are n random variables, each X_i can assume r_i states, and there are q_j combinations of parent states $\mathbf{Pa}_i$, then there are

$$|\theta| = \sum_{i=1}^{n} r_i \, q_i \tag{4.9}$$

total parameters. The parameters θ and the graph $\mathcal{G}$ fully specify the Bayesian network.

One of the primary tasks we seek to do with a Bayesian network is to perform **inference**: what is the probability of observing an instantiation $x_1, x_2, \ldots x_n$ of the net? A fundamental property of Bayesian networks is that this probability factors into local distributions:

$$p(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} p(X_i \mid \mathbf{Pa}_i). \tag{4.10}$$

This factorization permits fast and efficient computations to perform inference.

There are several benefits to using Bayesian networks in our problem setting:

(1) **Missing Data**: Bayesian networks admit a Bayesian approach to missing data by way of belief updating on unobserved nodes. In our case, order 50% of the data is unobserved in any given observations, even for observed nodes.

(2) **Bayesian Reasoning**: There are well-developed inference, prediction, and learning algorithms for Bayesian networks, all compatible with missing data in the Bayesian sense.

(3) **Interpretability**: Bayesian networks do not suffer from the "black-box" problem of interpretability. In our case this has tangible value, as we wish to provide human-interpretable reasons as to why an observation is predicted to be malicious.

(4) **Expert Knowledge**: The structure of the network can be elicited from domain experts, even in the absence of data.

This last point is crucial for us, as we will use the constructed net in an iterative fashion to obtain labeled data. As we discussed in the introduction, it is at first very difficult for an expert to

assess whether the raw data they see is associated with malicious activity. Constructing the network, including identifying both (1) which nodes to include and (2) identifying their relationships to each other, provides a means by which the expert can successfully provide labels T. Given the states of the observed nodes the expert labels the observation, which is then used to train the network; in particular to update estimates of the parameters for the hidden nodes. We discuss our procedure for parameter learning next.

4.3.1 Parameter Learning in Bayesian Networks

Typically we must estimate the parameters θ because they are unknown, and there are a lot of machine learning algorithms designed to do this under a variety of conditions and assumptions [100, 101, 102, 103, 104, 105, 104, 106, 107]. Each datapoint $\mathbf{y} \in \mathcal{X}$ is an **observation** consisting of some states of the random variables $\{X_i\}$ that were observed together. The simplest methods to learn θ asusme that the observations $y \in \mathcal{X}$ are **complete**, meaning that every node X_i has a value assigned to it. In this

4.3.2 Sensitivity Analysis in Bayesian Networks

Sensitivity analysis in Bayesian networks studies how much the model changes under small perturbations to parameter values. In this section we present a specific formulation of sensitivity and describe an application of this formulation to identify a subset of model parameters that are the most important (in a precise sense), following [108]. The motivation for this is that typically there are too many parameters (equation (4.9)) and not enough labeled data to learn them all effectively. In the active learning scenario which we describe below, we will seek a way to choose which datapoints for which we want to obtain labels; the sensitivity analysis presented here provides a mechanism to make this choice.

Given evidence e, a query y, and a probability parameter $x \in \theta$, then the posterior probability

as a function of x can be expressed as

$$p(y \mid e)(x) = \frac{\alpha x + \beta}{\gamma x + 1} \tag{4.11}$$

so the partial derivative is

$$\frac{\partial p(y \mid e)}{\partial x} = \frac{\alpha - \beta\gamma}{(\gamma x + 1)^2}. \tag{4.12}$$

The values of α, β and γ can be found by fixing three values of x, making an inference computation for each, and solving the resulting system of equations. This motivates the following definition of **parameter sensitivity**:

$$S(x \mid y, e) = \frac{\partial p(y \mid e)}{\partial x}. \tag{4.13}$$

S, in turn, can be used to define the concept of **parameter importance**:

$$I(x) = \frac{1}{mk} \sum_{y,e} S(x \mid y, e) \tag{4.14}$$

where m is the number of queries and k is the number of evidence scenarios.

The idea is then to find I for each parameter x and then identify the subset $\{x \in \theta \mid I(x) > \delta\}$ of "important" parameters for some threshold δ. There are two difficulties with this approach. First, the inference computations needed to find α, β and γ in equation (4.11) require estimates of θ. Our goal is to identify which parameters in θ are the most important to learn, and so we must have a way to provide reasonable estimates in the absence of any learned parameters in the first place. The second difficulty is that the importance as defined in equation (4.14) is expensive to compute: we must make three inference computations per evaluation of S, which itself must be computed mk times. Both m (the number of queries) and k (the number of evidence scenarios) grow exponentially with the size of the network. This is prohibitive for even relatively small networks.

We are presently working to find ways to resolve both of these difficulties in order to apply this approach to our work. We have currently implemented parameter bootstrapping methods to resolve the first difficulty (unknown parameters). To resolve the scaling obstacle we currently rely

on a related concept of sensitivity, the **mutual information** between two nodes in a network [97]:

$$I(X,Y) = \sum_{y,x} p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right) \tag{4.15}$$

This does not give as detailed an analysis as the preceding sensitivity results, as we obtain instead a measure of the mutual information between a target node and every other node in the network, instead of a per-parameter measure of importance, but it is computationally tractable. Using this approach we can find which nodes will inform the posterior on the target node the most, and focus our labeling efforts on those specific nodes.

4.3.3 Online Learning

In the standard classification task the goal is to learn a model $f : X \mapsto Y$ where X is the set of observations and Y are the labels. It is almost always the case in practice, however, that not all of X is known at the time of training. A simple approach is to regularly increase the size of the training set and retrain the model from scratch periodically. This is impractical when the dataset grows too large (in particular in the streaming limit). Moreover, this does not account for time-dependence in the underlying true distribution. In such cases we wish for the memory of the classifier to "fade" as new data become available.

Learning schemes that incorporate new data in a streaming fashion are known as **online learning** mechanisms. There are well-established protocols to do so in the specific case of Bayesian networks. Here our development follows [103]. The idea is as follows. We have a parameterized Bayesian network with parameters θ, where we write θ_{ijk} as defined in equation (4.8). We have a current estimate $\bar{\theta}$, possibly obtained from previous data, and we wish to update $\bar{\theta}$ in light of a set of new partial observations $D = \{y_1, y_2, \ldots, y_N\}$. The normalized data log likelihood is

$$\ell_D(\theta) = \frac{1}{N} \sum_{l=1}^{N} \log P(y_l \mid \theta). \tag{4.16}$$

Define the objective function

$$F(\theta) = \eta \ell_D(\theta) - d(\bar{\theta}, \theta) \tag{4.17}$$

where d is some distance on the parameter space and η is the learning rate (to be defined). The goal is then to find

$$\theta' = \arg\max_{\theta} F(\theta) \tag{4.18}$$

under the constraint

$$\sum_{k} \theta'_{ijk} = 1. \tag{4.19}$$

In the absence of d, this is equivalent to finding the maximum likelihood estimators for ℓ_D. The distance d is introduced to penalize updates that cause a large change in $\bar{\theta}$; a reflection of our confidence in the existing parameters. The learning rate η is a tunable parameter that sets how much we want to weight improving the data log likelihood given the new training data relative to penalizing changes as captured by d.

The first-order Taylor expansion of ℓ_D about $\bar{\theta}$ is given by

$$\ell_D(\theta) \simeq \ell_D(\bar{\theta}) + \nabla \ell_D(\bar{\theta}) \cdot (\theta - \bar{\theta}) \tag{4.20}$$

and the gradient is [109]

$$\nabla_{ijk} \ell_D(\theta) = \frac{\mathcal{E}_\theta(x_i^k, \mathbf{pa}_i^j \mid D)}{\theta_{ijk}} \tag{4.21}$$

where

$$\mathcal{E}_\theta(x_i^k, \mathbf{pa}_i^j \mid D) = \frac{1}{N} \sum_{l=1}^{N} p_\theta(x_i^k, \mathbf{pa}_i^j \mid y_l) \tag{4.22}$$

is the sample-based average; i.e., the average probability of observing states $x_i^k, \mathbf{pa}_i^j$ given y_l where the average is taken over D.

In [103] three choices for d are discussed: ℓ^2 norm, KL-divergence, and χ^2. Here we state only the result for χ^2, which is the one we will use. In this case, the solution to equation (4.18) is given by

$$\theta_{ijk} = \eta \frac{\mathcal{E}_{\bar{\theta}}(x_i^k, \mathbf{pa}_i^j \mid D)}{\mathcal{E}_{\bar{\theta}}(\mathbf{pa}_i^j \mid D)} + (1 - \eta)\bar{\theta}_{ijk}. \tag{4.23}$$

The hyperparameter η sets how much we want to update the parameters to fit newly observed data; the choice $\eta = 1$ will eliminate any previous learning by completely discarding $\bar{\theta}$ whereas $\eta = 0$ will leave $\bar{\theta}$ unchanged.

In practice we will use the update formula in equation 4.23 in two settings: for online learning, when we observe a batch of new labeled and verified data, and also when training the model the first time following the local learning phase.

4.4 Active Learning

Active learning is a subfield of machine learning in which the learning process is able to query an external source for information [81]. Active learning is motivated by the scenario in which an excess of unlabeled data is available but obtaining labels for these data is prohibitively expensive. Goals of active learning include optimizing the acquisition process itself and optimally choosing which data to label. Most research focuses on the latter, with the acquisition process typically treated as a symbolic query to a conditional distribution $p(y|x)$ asking for a label y given evidence x [110]. The act of obtaining such a label is referred to as **querying the oracle**.

Often the queries are expensive and difficult, typically when the oracle is a human expert and the labeling process requires expert knowledge. In this circumstance, the active learning setting is closely related to the fields of **knowledge engineering** [83] and **expert systems** [84]. Broadly, these fields deal with the creation of software that emulates human reasoning and decision-making. For our purposes, we take active learning to be the component of the knowledge engineering process focusing specifically on the acquistion of labeled data for use in machine learning algorithms. In particular, we are concerned with (1) which data should get labeled and (2) how to efficiently and accurately obtain labels for a given set of data.

Our learning and prediction framework, Bayesian networks, was one of the first to be applied to this task in the genesis of expert systems and knowledge engineering. We begin by addressing the second question: how to efficiently obtain labels. In Section 4.2 we introduced several potential approaches to this problem and discussed obstacles to using them in our case. We now introduce the approach we have taken, in the context of active learning: oracle epiphany.

4.4.1 Oracle Epiphany

One of the major goals of active learning is to identify which observations to label next. The multitude of ways to make this decision are often motivated by identifying observations that, when used to update parameter values, will optimize some cost function formulated in terms of concepts like mutual information or distance metrics on distribution spaces. For example, in Bayesian networks we might choose a new observation based on the sensitivity analysis we presented in Section 4.3.2 above.

These approaches do not take into account how difficult it is to obtain an answer for a given query. In our case the oracle queries highly non-uniform in their difficulty. Some are very easy, and some are too difficult to answer at all with the given information. Sometimes, however, an observation in isolation may be too difficult for the oracle to label, but once the oracle sees enough similar observations, her cognitive concept of the type of observation shifts and she is suddenly able to label all such observations at once. This phenomenon was introduced in [111] as **concept evolution** and formalized in [82] as **oracle epiphany**.

4.5 Proposed Method

We apply the concept of oracle epiphany to a specific data labeling task in the domain of cybersecurity. The dataset we use contains sensitive information and is the intellectual property of Respond Software, and as such we obscure specific information and details when necessary. The dataset corresponds to logs and telemetry used to provide Web filtering security and monitoring. Original, raw fields contain content such as time stamps and source and destination IP addresses; however, in this analysis we do not have access to much of this raw data as it has been redacted for privacy. This is consistent with the production environment, where such information would also be obscured.

4.5.1 Data

The original data consist of 224,724 events recorded over a 4 week time-span. Our ultimate goal is to construct a classifier than can identify which events correspond to malicious activity, such as the delivery of malware or spyware. The particulars of what constitutes malicious activity and how to identify it are within the purview of cyber security and outside of the scope of this thesis. Instead, our focus is on how to use oracle epiphany in conjunction with Bayesian networks in order to create a classifier in the face of numerous statistical challenges.

In this section our focus is specifically on the acquisition of labeled data. We assume we have access to an infinite supply of unlabeled datapoints $X \sim D_\theta$ where D_θ is a parametric model and X is a vector of categorical random variables. We assume the form of the distribution is given by a Bayesian network with known structure, but the parameters θ are unknown. Our goals are:

(1) Identify a subset of data $\mathcal{X} = \{x_1, x_2, \ldots x_n\}$ for which to obtain labels

(2) Obtain the labels $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$ where $T_i \in \{0, 1\}$ via oracle queries

(3) Use the labels and data to estimate the model parameters θ

This is ongoing work, and as a practical matter we are currently iterating between steps (1) and (2). Our results and discussion will therefore focus mainly on the acquisition of a labeled dataset, with preliminary results on training the model D_θ.

We begin with no labeled data. The fundamental problem is cyclic: in order to apply the principles of active learning we need to be able to compute (pseudo-)distances between conditional distributions, but in order to do this we need estimates of θ. Furthermore, the nature of the data is such that it is almost impossible to obtain a label for an isolated datapoint x_i.

4.5.2 Bayesian Network Representation

The Bayesian network we use is shown in Figure (4.3). There are a total of 39 active nodes (a few depicted are not currently implemented), of which 31 are observed and 8, including the target,

Table 4.1: Positive targets in labeled dataset.

	Featurized	**(ratio)**	**Raw**	**(ratio)**
Positive	8	0.0162	25	0.0001
Negative	475	0.9615	211332	0.9404
Unlabeled	27	0.0547	13367	0.0595
Total	494	1.0000	224724	1.0000

are hidden.

4.5.3 Ameliorating Class Imbalance Through Epiphany

Using the epiphany data acquisition method improves, but does not eliminate, the class imbalance problem. Recall that we have a binary target $T \in \{0, 1\}$ with $P(T = 1) \ll P(T = 0)$. Tabel (4.1) shows the rate of positive targets in the original dataset and as acquired through the epiphany method. Out of 224,724 raw events, only 25 were identified as positive, for a positive rate of 1.11×10^{-4}. Through the epiphany labeling process this rate is improved almost 100-fold: there are 8 positives out of 494 total observations, for a rate of 0.0162.

This rate is still in the regime of extreme class imbalance. Epiphany dramatically reduces the number of datapoints, from over 220 thousand to under 500, and improves the positive rate 100-fold, but does so at the cost of positive count in the observations: there are only 8 positive observations in the epiphany dataset. We therefore still need to use learning methods in the spirit of one-shot learning. We discuss learning from such data in Section 4.5.5 below. Before doing so we examine the labeled dataset in more detail.

4.5.4 Dataset Diversity

We first discuss the featurized data. Generally we still observe the problems of (i) high rates of missing data, (ii) extreme class imbalance on most nodes, and (iii) low variability, though the situation is improved in the featurized case. Figure (4.4a) shows the frequency of modal states in the featurized domain for the training data. Each column corresponds to a node. Within a

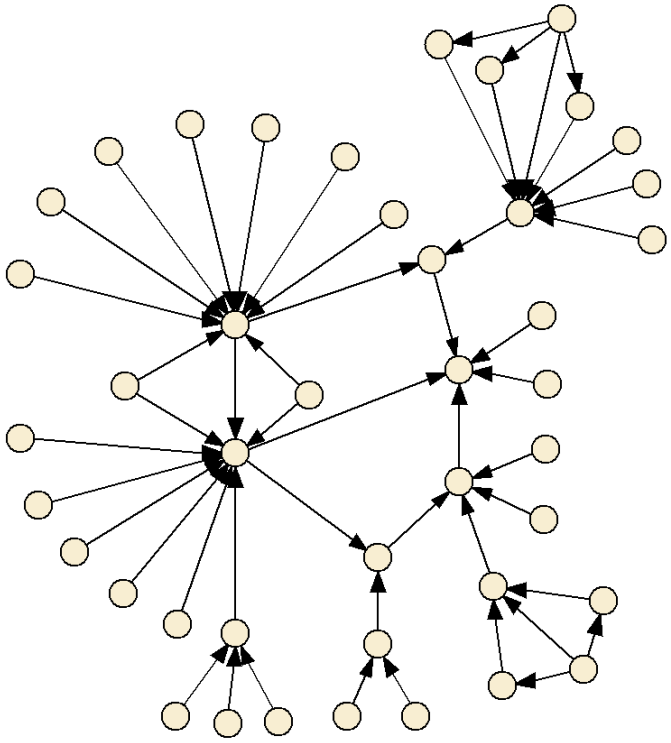


Figure 4.3: Structure of our Bayesian network.

Table 4.2: Missing data rates in featurized and raw data.

	Featurized	**Raw**
All	0.42	0.53
Observable	0.50	0.63
Hidden	0.03	0.06

column, we show the relative frequency with which that node is [i] unobserved (dark gray), [ii] observed in its most common (modal) state (light gray), and [iii] is observed in any other state (light blue). All node are categorical and many are binary; for those that are, the light blue denotes the frequency of the minority class. The nodes (columns) are sorted by the frequency with which they are unobserved.

The average rate at which nodes are unobserved, across all nodes and all observations, is 0.42 for this data (Table (4.2)), which is fairly high (in the figure this corresponds to the fraction of the plot filled by dark gray). This is one of the reasons a Bayesian approach is of value, since given enough data we can obtain accurate priors to use when a node is not observed. The amount of variation in the data is also low: the fraction of observations in a minority state is 0.068 (Table (4.2)). Figure (4.4b) shows the same data with unobserved data excluded to emphasize the relative frequency of minority classes.

An advantage of the oracle epiphany approach is that most of the hidden nodes in the training set are labeled. This is shown in Figure (4.5), where the hidden nodes are grouped in purple on the right. The expert was able to label almost all of the data: the fraction of unlabeled hidden nodes is 0.03 as opposed to 0.5 for observable nodes (Table (4.2)). The extreme class imbalance, however, is even more pronounced on hidden nodes: the rate of minority classes is 0.014 as opposed to 0.081

Table 4.3: Minority class rates in featurized and raw data.

	Featurized	**Raw**
All	0.068	0.021
Observable	0.081	0.026
Hidden	0.014	1.5E-4

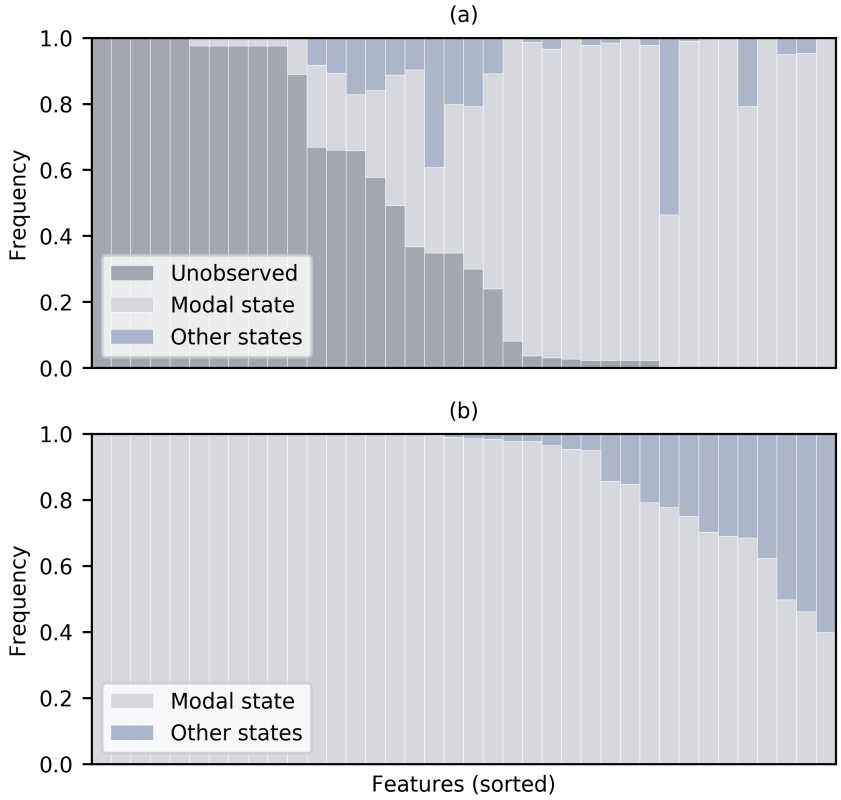


Figure 4.4: Frequency of modal states. Each column corresponds to one of 39 features, depicting the relative frequency of observing the modal state (light blue) as opposed to all other states (dark blue). (a) including unobserved features, sorted by frequency of unobserved data, (b) excluding unobserved features, sorted by frequency of modal state.

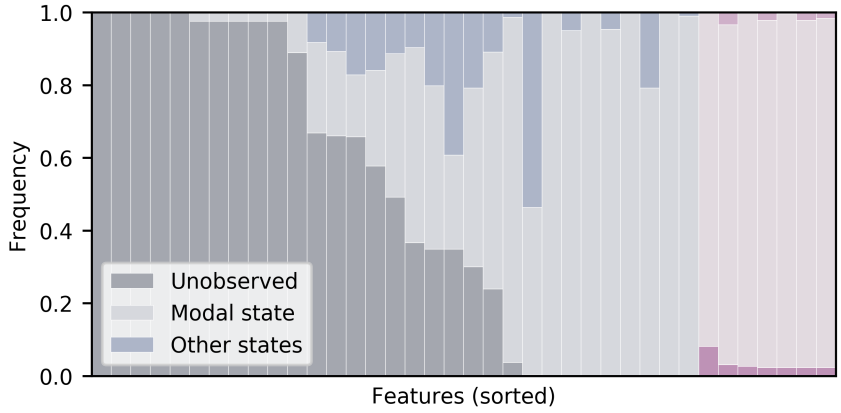


Figure 4.5: Frequency of modal states. Each column corresponds to one of 39 features, depicting the relative frequency of observing the modal state as opposed to all other states. Left (blue): observed variables. Right (purple): hidden variables.

for observable nodes (Table (4.3)).

Finally, Figure (4.6) shows the effect of featurization on dataset diversity by removing it. Here the counts are weighted by the number of events that generated the observation. Summary statistics for this case are presented in Tables 4.2 and 4.3, second column.

4.5.5 Learning and Classification

We now present results on our classifier accuracy. We use the simple cross-validation strategy of holding out 20% of the data for the test set, stratified by the target node T. Note that this means that we expect only 2 positive examples in the test set. Table (4.4) shows the performance on the training set. There are no false positives and only one false negative. Table (4.5) shows performance on the test set - here there are no errors at all. Because the dataset is so small ($N = 494$) and there are so few positive examples ($N_{pos} = 8$) it is difficult to draw meaningful conclusions from these results. At a minimum, however, the findings are indicative that there is simpler underlying structure to the dataset.

If the features were continuous we could apply methods similar to those presented in Chapter 3, or even simpler dimensionality reduction techniques like an SVD, to find a lower-dimensional

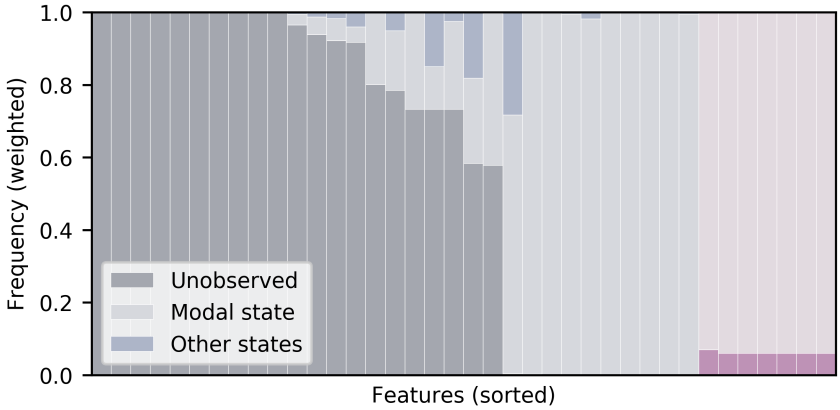


Figure 4.6: Frequency of modal states weighted by occurrence. Each column corresponds to one of 39 features, depicting the relative frequency of observing the modal state as opposed to all other states. Left (blue): observed variables. Right (purple): hidden variables.

Table 4.4: Confusion matrix, training set.

	True Positive	**True Negative**
Predicted Positive	5	1
Predicted Negative	0	356

space that still captures the variability in the data. If we could transform the data into this space efficiently (e.g. random projection style) then we could combine this with the oracle epiphany featurization used here to shrink the data in both latent dimension and number of observations. Unfortunately it is not obvious how to perform such transformations on categorical data. There are, however, some approaches available, which we are in the process of implementing (see Section (4.6) for details).

We predict $T = 1$ for an observations $\mathbf{e}$ if $p(T = 1 \mid \mathbf{e}) > 0.5$. Since there is only one error across both the training and test set, it is natural to investigate the empirical distribution of T. This is shown in figure (4.7), for both (a) the featurized data and (b) the raw data. In (a) we see that the vast majority of observations have $p(T = 1 \mid \mathbf{e}) < 0.05$ (note that the y-axis is a log scale), with a few other non-zero bins. Importantly, there are no observations in the range $0.25 < p(T = 1 \mid \mathbf{e}) < 0.75$. Taking these probabilities as a measure of the **confidence** of the model, we see that there are no "hard" cases for the model, it is always very confident about T.

Figure (4.7b) shows the same distribution over the raw data. Out of 224,724 events only 30 have $p(T = 1 \mid \mathbf{e}) > 0.002$ (not enough to have visible bars in the log plot shown here). The oracle epiphany featurization process thus leads to greater variability in $p(T = 1 \mid \mathbf{e})$, though this distribution is still highly-peaked and strongly suggestive of a simpler underlying structure.

Table 4.5: Confusion matrix, test set.

	True Positive	**True Negative**
Predicted Positive	2	0
Predicted Negative	0	119

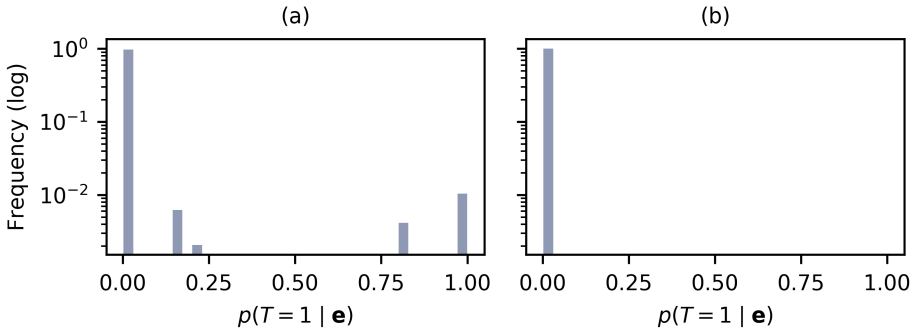


Figure 4.7: Probability that T=1 over X for (a) featurized and (b) raw data.

4.6 Conclusions

By applying Oracle epiphany to both the feature engineering and labeling tasks we reduced the number of data points to consider from 2.2×10^5 down to 494. We also increased the relative rate of the minority class by order 100, from 0.0001 to 0.0162. We trained a Bayesian network classifier on the labeled data to obtain promising preliminary results for anomaly detection. This work is ongoing, and we are currently in the process of refining the epiphany process as well as obtaining more labeled data. In the short term we aim to identify more minority class observations manually, though we intend to pursue some type of cluster analysis on unlabeled data in order to facilitate this process in the near future. We will also investigate other approaches to anomaly detection. An interesting mathematical consideration is whether the bounds in [82] for general Oracle epiphanies can be adapted to our specific case.

Bibliography

[1] E. P. Kightley, A. Pearson, J. A. Evans, and D. M. Bortz, "Fragmentation of biofilm-seeded bacterial aggregates in shear flow," *European Journal of Applied Mathematics*, vol. 29, no. 6, pp. 1062–1078, 2018.

[2] E. P. Kightley and S. Becker, "One-Pass Sparsified Gaussian Mixtures," *arXiv:1903.04056*, 2019.

[3] C. Binnie, M. Kimber, and H. Thomas, "Coagulation and flocculation," in *Basic Water Treatment*, pp. 61–83, ICE Publishing, 2017.

[4] D. N. Thomas, S. J. Judd, and N. Fawcett, "Flocculation modelling: A review," *Water Research*, vol. 33, no. 7, pp. 1579–1592, 1999.

[5] K. J. Verstrepen and F. M. Klis, "Flocculation, adhesion and biofilm formation in yeasts," *Molecular Microbiology*, vol. 60, no. 1, pp. 5–15, 2006.

[6] E. V. Soares, "Flocculation in Saccharomyces cerevisiae: A review," *Journal of Applied Microbiology*, vol. 110, no. 1, pp. 1–18, 2011.

[7] F. Gassara, C. Antzak, C. M. Ajila, S. J. Sarma, S. K. Brar, and M. Verma, "Chitin and chitosan as natural flocculants for beer clarification," *Journal of Food Engineering*, vol. 166, pp. 80–85, 2015.

[8] G. Stewart, "Saccharomyces species in the Production of Beer," *Beverages*, vol. 2, no. 4, p. 34, 2016.

[9] V. Vidgren and J. Londesborough, "125th anniversary review: Yeast flocculation and sedimentation in brewing," *Journal of the Institute of Brewing*, vol. 117, no. 4, pp. 475–487, 2011.

[10] G. S. Dhillon, S. Kaur, S. K. Brar, and M. Verma, "Flocculation and haze removal from crude beer using in-house produced laccase from Trametes versicolor cultured on brewers spent grain," *Journal of Agricultural and Food Chemistry*, vol. 60, no. 32, pp. 7895–7904, 2012.

[11] A. K. Verma, R. R. Dash, and P. Bhunia, "A review on chemical coagulation/flocculation technologies for removal of colour from textile wastewaters," *Journal of Environmental Management*, vol. 93, no. 1, pp. 154–168, 2012.

[12] C. S. Lee, J. Robinson, and M. F. Chong, "A review on application of flocculants in wastewater treatment," *Process Safety and Environmental Protection*, vol. 92, no. 6, pp. 489–508, 2014.

[13] S. Suarez, J. M. Lema, and F. Omil, "Pre-treatment of hospital wastewater by coagulation-flocculation and flotation," *Bioresource Technology*, vol. 100, no. 7, pp. 2138–2146, 2009.

[14] J. J. Milledge and S. Heaven, "A review of the harvesting of micro-algae for biofuel production," *Reviews in Environmental Science and Biotechnology*, vol. 12, no. 2, pp. 165–178, 2013.

[15] S. B. Ummalyma, E. Gnansounou, R. K. Sukumaran, R. Sindhu, A. Pandey, and D. Sahoo, "Bioflocculation: An alternative strategy for harvesting of microalgae – An overview," *Bioresource Technology*, vol. 242, pp. 227–235, 2017.

[16] D. Wrede, K. Kadali, T. Stevenson, A. S. Ball, A. F. Miranda, M. Taha, and A. Mouradov, "Co-Cultivation of Fungal and Microalgal Cells as an Efficient System for Harvesting Microalgal Cells, Lipid Production and Wastewater Treatment," *PLoS ONE*, vol. 9, no. 11, p. e113497, 2014.

[17] A. S. Japar, M. S. Takriff, and N. H. M. Yasin, "Harvesting microalgal biomass and lipid extraction for potential biofuel production: A review," 2017.

[18] Y. Yuan and R. R. Farnood, "Strength and breakage of activated sludge flocs," *Powder Technology*, vol. 199, no. 2, pp. 111–119, 2010.

[19] P. Jarvis, B. Jefferson, J. Gregory, and S. A. Parsons, "A review of floc strength and breakage," *Water Research*, vol. 39, no. 14, pp. 3121–3137, 2005.

[20] G. H. Yu, P. J. He, L. M. Shao, and D. J. Lee, "Enzyme activities in activated sludge flocs," *Applied Microbiology and Biotechnology*, vol. 77, no. 3, pp. 605–612, 2007.

[21] D. Snidaro, F. Zartarian, F. Jorand, J. Y. Bottero, J. C. Block, and J. Manem, "Characterization of activated sludge flocs structure," *Water Science and Technology*, vol. 36, no. 4, pp. 313–320, 1997.

[22] J. Bratby, *Coagulation and Flocculation in Water and Wastewater Treatment*, vol. 15. IWA Publishing, 2016.

[23] S. N. Liss, I. G. Droppo, G. G. Leppard, and T. G. Milliagn, *Flocculation in Natural and Engineered Environmental Systems*. CRC Press, 2007.

[24] I. Nopens, *Modelleren van het actief-slib flocculatieproces: een populatiebalansbenadering (Modelling the activated sludge flocculation process: a population balance approach)*. PhD thesis, Universiteit Gent, 2005.

[25] D. M. Bortz, T. L. Jackson, K. A. Taylor, A. P. Thompson, and J. G. Younger, "Klebsiella pneumoniae flocculation dynamics," *Bulletin of Mathematical Biology*, vol. 70, no. 3, pp. 745–768, 2008.

[26] I. Mirzaev and D. M. Bortz, "A numerical framework for computing steady states of structured population models and their stability," *Mathematical Biosciences and Engineering*, vol. 14, no. 4, pp. 933–952, 2017.

[27] J. Solsvik, S. Tangen, and H. A. Jakobsen, "On the constitutive equations for fluid particle breakage," *Reviews in Chemical Engineering*, vol. 29, no. 5, pp. 241–356, 2013.

[28] S. Blaser, "Flocs in shear and strain flows," *Journal of Colloid and Interface Science*, vol. 225, no. 2, pp. 273–284, 2000.

[29] D. James, N. Yogachandran, M. Loewen, H. Liu, and A. Davis, "Floc rupture in extensional flow," *Journal of Pulp and Paper Science*, vol. 29, no. 11, pp. 377–383, 2003.

[30] S. Blaser, "Break-up of flocs in contraction and swirling flows," *Colloids and Surfaces A: Physicochemical and Engineering Aspects*, vol. 166, no. 1-3, pp. 215–223, 2000.

[31] M. Kobayashi, "Strength of natural soil flocs," *Water Research*, vol. 39, no. 14, pp. 3273–3278, 2005.

[32] M. Kobayashi, "Breakup and strength of polystyrene latex flocs subjected to a converging flow," *Colloids and Surfaces A: Physicochemical and Engineering Aspects*, vol. 235, no. 1-3, pp. 73–78, 2004.

[33] E. Byrne, S. Dzul, M. Solomon, J. Younger, and D. M. Bortz, "Postfragmentation density function for bacterial aggregates in laminar flow," *Physical Review E*, vol. 83, no. 4, pp. 1–10, 2011.

[34] I. Mirzaev, E. C. Byrne, and D. M. Bortz, "An inverse problem for a class of conditional probability measure-dependent evolution equations," *Inverse Problems*, vol. 32, no. 9, p. 095005, 2016.

[35] J. A. Stotsky, J. F. Hammond, L. Pavlovsky, E. J. Stewart, J. G. Younger, M. J. Solomon, and D. M. Bortz, "Variable viscosity and density biofilm simulations using an immersed boundary method, part II: Experimental validation and the heterogeneous rheology-IBM," *Journal of Computational Physics*, vol. 317, pp. 204–222, 2016.

[36] N. E. Jackson and C. L. Tucker, "A model for large deformation of an ellipsoidal droplet with interfacial tension," *Journal of Rheology*, vol. 47, no. 3, pp. 659–682, 2003.

[37] E. D. Wetzel and C. L. Tucker, "Droplet deformation in dispersions with unequal viscosities and zero interfacial tension," *Journal of Fluid Mechanics*, vol. 426, pp. 199–228, jan 2001.

[38] S. Blaser, "Forces on the surface of small ellipsoidal particles immersed in a linear flow field," *Chemical Engineering Science*, vol. 57, no. 3, pp. 515–526, 2002.

[39] F. Gaboriaud, M. L. Gee, R. Strugnell, and J. F. Duval, "Coupled electrostatic, hydrodynamic, and mechanical properties of bacterial interfaces in aqueous media," *Langmuir*, vol. 24, no. 19, pp. 10988–10995, 2008.

[40] M. Kleman and O. D. Lavrentovich, *Soft Matter Physics: an Introduction.* New York: Springer, 2003.

[41] J. F. Hammond, E. J. Stewart, J. G. Younger, M. J. Solomon, and D. M. Bortz, "Variable Viscosity and Density Biofilm Simulations using an Immersed Boundary Method, Part I: Numerical Scheme and Convergence Results," *Computer Modeling in Engineering Sciences*, vol. 98, no. 3, pp. 295–340, 2014.

[42] C. J. Rueb and C. F. Zukoski, "Viscoelastic properties of colloidal gels," *Journal of Rheology*, vol. 41, no. 2, pp. 197–218, 1997.

[43] A. L. Yarin, O. Gottlieb, and I. V. Roisman, "Chaotic rotation of triaxial ellipsoids in simple shear flow," *Journal of Fluid Mechanics*, vol. 340, p. S0022112097005260, jun 1997.

[44] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[45] S. Muthukrishnan, *Data Streams: Algorithms and Applications*. Breda, The Netherlands: Now Publishers, 2005.

[46] F. Pourkamali-Anaraki and S. Becker, "Preconditioned Data Sparsification for Big Data with Applications to PCA and K-means," *IEEE Transactions on Information Theory*, vol. 63, no. 5, pp. 2954–2974, 2017.

[47] K. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[48] W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space," *Contemporary Mathematics*, vol. 26, pp. 189–206, 1984.

[49] N. Ailon and B. Chazelle, "The Fast Johnson-Lindenstrauss Transform and Approximate Nearest Neighbors," *SIAM Journal on Computing Comput*, vol. 39, no. 1, pp. 302–322, 2009.

[50] E. Candès and M. Wakin, "An Introduction To Compressive Sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[51] D. Achlioptas, "Database-friendly random projections: Johnson-Lindenstrauss with binary coins," *Journal of Computer and System Sciences*, vol. 66, no. 4, pp. 671–687, 2003.

[52] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Review*, vol. 53, no. 2, pp. 217–288, 2011.

[53] D. P. Woodruff, "Sketching as a Tool for Numerical Linear Algebra," *Foundations and Trends in Theoretical Computer Science*, vol. 10, no. 1-2, pp. 1–157, 2014.

[54] C. Boutsidis, A. Zouzias, M. W. Mahoney, and P. Drineas, "Randomized dimensionality reduction for κ-means clustering," *IEEE Transactions on Information Theory*, vol. 61, no. 2, pp. 1045–1062, 2015.

[55] M. B. Cohen, S. Elder, C. Musco, C. Musco, and M. Persu, "Dimensionality Reduction for k-Means Clustering and Low Rank Approximation," in *STOC'15*, pp. 163–172, 2015.

[56] H. Reboredo, F. Renna, R. Calderbank, and M. R. Rodrigues, "Projections designs for compressive classification," *2013 IEEE Global Conference on Signal and Information Processing, GlobalSIP 2013 - Proceedings*, pp. 1029–1032, 2013.

[57] S. Dasgupta, "Learning mixtures of Gaussians," in *Proc eedings of the 40th Annual IEEE Symposium Foundations of Computer Science*, pp. 634–644, 1999.

[58] S. Dasgupta and A. Gupta, "An Elementary Proof of a Theorem of Johnson and Lindenstrauss," *Random Structures and Algorithms*, vol. 22, no. 1, pp. 60–65, 2003.

[59] N. Linial, E. London, and Y. Rabinovich, "The geometry of graphs and some of its algorithmic applications," *Combinatorica*, vol. 15, no. 2, pp. 215–245, 1995.

[60] P. Indyk, "Stable distributions, pseudorandom generators, embeddings, and data stream computation," *Journal of the ACM*, vol. 53, no. 3, pp. 307–323, 2006.

[61] L. J. Schulman, "Clustering for edge-cost minimization," in *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing (STOC)*, (New York), pp. 547–555, ACM, 2000.

[62] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala, "Latent semantic indexing: A probabilistic analysis," *Journal of Computer and System Sciences*, vol. 61, no. 2, pp. 217–235, 2000.

[63] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing (STOC)*, (New York), pp. 604–613, ACM, 1998.

[64] E. Kushilevitz, R. Ostrovsky, and Y. Rabani, "Efficient Search for Approximate Nearest Neighbor in High Dimensional Spaces," *SIAM Journal on Computing*, vol. 30, no. 2, pp. 457–474, 2003.

[65] P. Frankl and H. Maehara, "The Johnson-Lindenstrauss lemma and the sphericity of some graphs," *Journal of Combinatorial Theory, Series B*, vol. 44, no. 3, pp. 355–362, 1988.

[66] N. Alon, "Problems and results in extremal combinatorics—I," *Discrete Mathematics*, vol. 273, pp. 31–53, 2003.

[67] M. W. Mahoney, "Randomized algorithms for matrices and data," *Foundations and Trends in Machine Learning*, vol. 3, no. 2, pp. 123–224, 2011.

[68] J. A. Tropp, "An Introduction to Matrix Concentration Inequalities," *Foundations and Trends in Machine Learning*, vol. 8, no. 1-2, pp. 1–230, 2015.

[69] T. T. Do, L. Gan, N. H. Nguyen, and T. D. Tran, "Fast and efficient compressive sensing using structurally random matrices," *IEEE Transactions on Signal Processing*, vol. 60, no. 1, pp. 139–154, 2012.

[70] R. L. Streit, *Poisson Point Processes*. Boston, MA: Springer, 2019.

[71] A. P. Dempster, N. Laird, and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[72] R. Sundberg, *Maximum likelihood theory and applications for distributions generated when observing a function of an exponential family variable*. PhD thesis, Stockholm University, 1971.

[73] R. Sundberg, "Maximum Likelihood Theory for Incomplete Data from an Exponential Family," *Scandinavian Journal of Statistics*, vol. 1, no. 2, pp. 49–58, 1974.

[74] R. Sundberg, "An Iterative Method for Solution of the Likelihood Equations for Incomplete Data from Exponential Families," *Communications in Statistics - Simulation and Computation*, vol. 5, no. 1, pp. 55–64, 1976.

[75] C. Wu and F. Jeff, "On the Convergence Properties of the EM Algorithm," *The Annals of Statistics*, vol. 11, no. 1, pp. 95–103, 1983.

[76] G. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*. Wiley-Interscience, second ed., 2008.

[77] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge: Cambridge University Press, first ed., 2004.

[78] K. B. Petersen and M. S. Pedersen, *The Matrix Cookbook*. Technical University of Denmark, 2012.

[79] D. Arthur and S. Vassilvitskii, "K-Means++: the Advantages of Careful Seeding," *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, vol. 8, pp. 1027–1025, 2007.

[80] C. E. Rasmussen and C. K. I. Williams, *Guassian Processes for Machine Learning*. Cambridge, MA: MIT Press, 2005.

[81] B. Settles, "Active Learning Literature Survey," tech. rep., University of Wisconsin-Madison, 2010.

[82] T.-K. Huang, L. Li, A. Vartanian, S. Amershi, and X. Zhu, "Active Learning with Oracle Epiphany," in *30th Conference on Neural Information Processing Systems (NIPS 2016)*, 2016.

[83] R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge engineering: principles and methods," *Data and Knowledge Engineering*, vol. 25, pp. 161–197, 1998.

[84] S. H. Liao, "Expert system methodologies and applications-a decade review from 1995 to 2004," *Expert Systems with Applications*, vol. 28, no. 1, pp. 93–103, 2005.

[85] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[86] E. Rodner and J. Denzler, "One-shot learning of object categories using dependent Gaussian processes," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6376 LNCS, pp. 232–241, 2010.

[87] L. Fei-Fei, R. Fergus, and P. Perona, "A Bayesian approach to unsupervised one-shot learning of object categories," in *9th IEEE International Conference on Computer Vision*, 2003.

[88] H. Altae-Tran, B. Ramsundar, A. S. Pappu, and V. Pande, "Low Data Drug Discovery with One-Shot Learning," *ACS Central Science*, vol. 3, no. 4, pp. 283–293, 2017.

[89] M. G. Lagoudakis and R. Parr, "Reinforcement Learning as Classification," in *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, 2003.

[90] J. Janisch, T. Pevný, and V. Lisý, "Classification with Costly Features using Deep Reinforcement Learning," *arXiv:1711.07364v2*, 2019.

[91] M. A. Wiering, H. Van Hasselt, A. D. Pietersma, and L. Schomaker, "Reinforcement learning algorithms for solving classification problems," in *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, 2011.

[92] E. Lin, Q. Chen, and X. Qi, "Deep Reinforcement Learning for Imbalanced Classification," *arXiv:1901.91379v1*, 2019.

[93] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[94] T. Hubert, J. Schrittwieser, L. Baker, F. Hui, D. Hassabis, I. Antonoglou, D. Silver, A. Bolton, K. Simonyan, L. Sifre, T. Lillicrap, T. Graepel, G. van den Driessche, Y. Chen, M. Lai, A. Huang, and A. Guez, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[95] D. Kumaran, D. Hassabis, T. Graepel, M. Lai, D. Silver, M. Lanctot, L. Sifre, T. Hubert, K. Simonyan, I. Antonoglou, T. Lillicrap, J. Schrittwieser, and A. Guez, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[96] J. Pearl, "Bayesian Networks: a Model of Self-Activated Memory for Evidential Reasoning," in *Proceedings of the 7th Conference of the Cognitive Science Society*, University of California, Irvine CA, 1985.

[97] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. San Francisco: Morgan Kaufmann, 1988.

[98] R. E. Neapolitan, *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*. Wiley, 1989.

[99] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

[100] F. R. Bach and M. I. Jordan, "Learning graphical models for stationary time series," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2189–2199, 2004.

[101] N. Angelopoulos and J. Cussens, "Bayesian learning of Bayesian networks with informative priors," *Annals of Mathematics and Artificial Intelligence*, vol. 54, no. 1-3, pp. 53–98, 2009.

[102] L. Jinzhong and L. Qin, "Online learning of Bayesian network parameters," in *4th International Conference on Natural Computation, ICNC 2008*, vol. 3, pp. 267–271, 2008.

[103] E. Bauer, D. Koller, and Y. Singer, "Update Rules for Parameter Estimation in Bayesian Networks," in *Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI1997)*, pp. 3–13, 1997.

[104] S. Lim and S.-B. Cho, "Online Learning of Bayesian Network Parameters with Incomplete Data," in *International Conference on Intelligent Computing (ICIC 2006)*, vol. 55, pp. 309–314, 2006.

[105] L. Li, *Budgeted Parameter Learning of Generative Bayesian Networks*. PhD thesis, University of Alberta, 2009.

[106] D. Heckerman, D. Geiger, and D. M. Chickering, "Learning Bayesian Networks: The Combination of Knowledge and Statistical Data Metrics for Belief Networks," *Machine Learning*, vol. 20, no. 3, pp. 197–243, 1993.

[107] W. Liao and Q. Ji, "Learning Bayesian network parameters under incomplete data with domain knowledge," *Pattern Recognition*, vol. 42, no. 11, pp. 3046–3056, 2009.

[108] H. Wang, I. Rish, and S. Ma, "Using sensitivity analysis for selective parameter update in Bayesian network learning," *In Proceedings of the AAAI spring symposium on information refinement and revision for decision making: Modeling for diagnostics, prognostics and prediction*, pp. 29–36, 2002.

[109] J. Binder, D. Koller, S. Russell, and K. Kanazawa, "Addaptive Probabilistic Networks with Hidden Variables," *Machine Learning*, vol. 29, no. 2-3, pp. 213–244, 1997.

[110] Y. Yang and M. Loog, "Single shot active learning using pseudo annotators," *Pattern Recognition*, vol. 89, pp. 22–31, 2019.

[111] T. Kulesza, S. Amershi, R. Caruana, D. Fisher, and D. Charles, "Structured labeling for facilitating concept evolution in machine learning," in *CHI*, pp. 3075–3084, 2014.