



AFRL-RY-WP-TR-2020-0298

ONLINE CLUSTERING WITH BAYESIAN NONPARAMETRICS

**Matthew D. Scherreik
Wright State University**

**OCTOBER 2020
Final Report**

Approved for public release; distribution is unlimited.

See additional restrictions described on inside pages

© 2020 Matthew D. Scherreik

STINFO COPY

**AIR FORCE RESEARCH LABORATORY
SENSORS DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7320
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE**

REPORT DOCUMENTATION PAGE					<i>Form Approved</i> OMB No. 0704-0188	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.						
1. REPORT DATE (DD-MM-YY) October 2020		2. REPORT TYPE Thesis		3. DATES COVERED (From - To) 5 October 2020 –5 October 2020		
4. TITLE AND SUBTITLE ONLINE CLUSTERING WITH BAYESIAN NONPARAMETRICS				5a. CONTRACT NUMBER In-house		
				5b. GRANT NUMBER		
				5c. PROGRAM ELEMENT NUMBER N/A		
6. AUTHOR(S) Matthew D. Scherreik				5d. PROJECT NUMBER N/A		
				5e. TASK NUMBER N/A		
				5f. WORK UNIT NUMBER N/A		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Wright State University 3640 Colonel Glenn Hwy. Dayton, OH 45435				8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory Sensors Directorate Wright-Patterson Air Force Base, OH 45433-7320 Air Force Materiel Command United States Air Force				10. SPONSORING/MONITORING AGENCY ACRONYM(S) AFRL/RWYE		
				11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S) AFRL-RY-WP-TR-2020-0298		
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.						
13. SUPPLEMENTARY NOTES PAO case number 88ABW-2020-3061, Clearance Date 5 October 2020. © 2020 Matthew D. Scherreik. Presented in partial fulfillment of the requirements for the degree of Ph.D. in Electrical Engineering to Wright State University. This work was funded in whole or in part by Department of the Air Force. The U.S. Government has for itself and others acting on its behalf an unlimited, paid-up, nonexclusive, irrevocable worldwide license to use, modify, reproduce, release, perform, display, or disclose the work by or on behalf of the U.S. Government. Report contains color.						
14. ABSTRACT Clustering algorithms, such as Gaussian mixture models and K-means, often require the number of clusters to be specified a priori. Bayesian nonparametric (BNP) methods avoid this problem by specifying a prior distribution over the cluster assignments that allows the number of clusters to be inferred from the data. This can be especially useful for online clustering tasks, where data arrives in a continuous stream and the number of clusters may dynamically change over time. Classical BNP priors often overestimate the number of clusters, however, leading researchers to develop new priors with more control over this tendency. To date, BNP algorithms resistant to over-clustering have only been implemented for offline processing, utilizing Markov chain Monte Carlo inference. In this dissertation, we derive a novel algorithm for online BNP clustering using variational inference, with explicit control over the over-clustering phenomenon. Additionally, we propose two methods for tuning a critical hyperparameter mid-stream, based on empirical analysis of the BNP cluster assignment prior and a cost function from Gaussian mixture reduction. We demonstrate the effectiveness of our algorithms on dynamic datasets designed specifically to challenge online BNP clustering algorithms. We also show that our algorithms can be employed for practical applications of radar pulse clustering and neural spike sorting, achieving competitive—and often superior—results when compared to classical BNP methods. Furthermore, we exploit the model-based framework to extend our algorithm and tuning methods from purely Gaussian mixtures to handle data with mixed multivariate Gaussian and categorical type, and demonstrate this new extension on real-world data. Our empirical studies indicate that the developments in this dissertation are a significant contribution to the state of the art in BNP clustering.						
15. SUBJECT TERMS clustering						
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT: SAR	18. NUMBER OF PAGES 134	19a. NAME OF RESPONSIBLE PERSON (Monitor) Matthew D. Scherreik 19b. TELEPHONE NUMBER (Include Area Code) N/A	
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified				

ONLINE CLUSTERING WITH BAYESIAN
NONPARAMETRICS; *OR* A BAYESIAN
MODEL-BASED METHOD FOR CLUSTERING
DYNAMIC DATASETS; *OR* MODEL-BASED
CLUSTERING FOR DYNAMIC DATA

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Ph.D. in Electrical Engineering

by

MATTHEW D. SCHERREIK
B.S.E.E., Wright State University, 2013
M.S.E.E., Wright State University, 2014

2020
Wright State University

Wright State University
COLLEGE OF GRADUATE STUDIES

August 28, 2020

I HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER MY SUPERVISION BY Matthew D. Scherreik ENTITLED Online Clustering with Bayesian Nonparametrics; OR A Bayesian Model-Based Method for Clustering Dynamic Datasets; OR Model-Based Clustering for Dynamic Data BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Ph.D. in Electrical Engineering.

Brian Rigling, Ph.D.
Dissertation Director

Arnab Shaw, Ph.D.
Director, Ph.D. in Electrical Engineering Program

Barry Milligan, Ph.D.
Interim Dean of the Graduate School

Committee on
Final Examination

Brian Rigling, Ph.D.

Fred Garber, Ph.D.

Arnab Shaw, Ph.D.

Joshua Ash, Ph.D.

John Gallagher, Ph.D.

ABSTRACT

Scherreik, Matthew. Ph.D. E.E., Department of Electrical Engineering, Wright State University, 2020. *Online Clustering with Bayesian Nonparametrics; OR A Bayesian Model-Based Method for Clustering Dynamic Datasets; OR Model-Based Clustering for Dynamic Data* .

Clustering algorithms, such as Gaussian mixture models and K-means, often require the number of clusters to be specified a priori. Bayesian nonparametric (BNP) methods avoid this problem by specifying a prior distribution over the cluster assignments that allows the number of clusters to be inferred from the data. This can be especially useful for online clustering tasks, where data arrives in a continuous stream and the number of clusters may dynamically change over time. Classical BNP priors often overestimate the number of clusters, however, leading researchers to develop new priors with more control over this tendency. To date, BNP algorithms resistant to over-clustering have only been implemented for offline processing, utilizing Markov chain Monte Carlo inference. In this dissertation, we derive a novel algorithm for online BNP clustering using variational inference, with explicit control over the over-clustering phenomenon. Additionally, we propose two methods for tuning a critical hyperparameter mid-stream, based on empirical analysis of the BNP cluster assignment prior and a cost function from Gaussian mixture reduction. We demonstrate the effectiveness of our algorithms on dynamic datasets designed specifically to challenge online BNP clustering algorithms. We also show that our algorithms can be employed for practical applications of radar pulse clustering and neural spike sorting, achieving competitive—and often superior—results when compared to classical BNP methods. Furthermore, we exploit the model-based framework to extend our algorithm and tuning methods from purely Gaussian mixtures to handle data with mixed multivariate Gaussian and categorical type, and demonstrate this new extension on real-world data. Our empirical studies indicate that the developments in this dissertation are a significant contribution to the state of the art in BNP clustering.

Contents

List of Figures	vi
List of Tables	vii
1 Introduction	1
1.1 Contributions	3
1.2 Published and submitted papers related to this dissertation	5
1.3 Outline	6
1.4 Notation	6
2 Background	8
2.1 Literature review	8
2.1.1 Brief taxonomy of clustering algorithms	9
2.1.2 Number of clusters	10
2.1.3 Offline vs. online clustering	11
2.1.4 Related work	12
2.2 Variational BNP clustering	13
2.2.1 Online variational BNP clustering	16
2.3 Summary	17
3 Algorithm derivation	19
3.1 Redundancy in BNP clustering	19
3.2 Reducing redundancy in online BNP clustering	20
3.3 Inference for multivariate Gaussian data	22
3.4 Remarks on implementation	25
3.5 Algorithm demonstration	27
3.6 Summary	29
4 Hyperparameter tuning	30
4.1 Empirical properties of the PCR-P hyperparameter	31
4.2 Numerical methods for PCR-P hyperparameter tuning	34
4.2.1 Gaussian mixture reduction	35
4.2.2 Analyzing NISE for redundant models	36

4.2.3	Employing the NISE to tune r	38
4.2.4	Selecting η	39
4.3	Combining inference-based and numerical approaches	40
4.4	Tuning the hyperparameter online	42
4.5	Summary	45
5	Experiments with synthetic data	46
5.1	Methodology and metrics	47
5.2	Rotating dataset	49
5.3	Radial dataset	51
5.4	Summary	56
6	Experiments with radar pulse data	58
6.1	Related work	60
6.2	Simulation	61
6.2.1	Methodology and metrics	67
6.3	Results	67
6.4	Summary	71
7	Neural spike sorting with BNP modeling	73
7.1	Experiments	74
7.1.1	Spike feature extraction via weighted PCA of wavelets	74
7.1.2	Results	75
7.2	Summary	78
8	Clustering data with mixed type	80
8.1	A BNP model for mixed data types	81
8.2	Computing NISE for mixed data models	84
8.3	Experiments with mixed-type data	87
8.3.1	Heart disease	89
8.3.2	Wine ratings	92
8.4	Summary	97
9	Conclusion	99
9.1	Contributions	99
9.2	Expected impact	100
9.3	Future work	101
	References	103

List of Figures

3.1	Cluster count estimation error ΔK for various values of r , α , and N_k on a six-cluster problem	28
4.1	Empirical distributions for K given r	32
4.2	Curve fitment for $(\bar{r}_i, \hat{\xi})$ pairs	33
4.3	NISE vs $\Delta K/K$ for redundant GMMs	37
4.4	Analysis of root-finding hyper-hyperparameter η	41
5.1	Example data from eight consecutive minibatches of a rotating dataset	50
5.2	Aggregated performance metrics on the rotating dataset	51
5.3	ΔK per minibatch on the rotating dataset	52
5.4	Example data from eight consecutive minibatches of a radial dataset	53
5.5	Aggregated performance metrics on the radial dataset	55
5.6	ΔK per minibatch on the radial dataset	56
6.1	Emitter layout for the radar scenarios	64
6.2	Pulse cluster shapes for the radar scenarios	65
6.3	Number of emitters within detection range as a function of time	66
6.4	Number of detected pulses as a function of time	66
6.5	Aggregated performance metrics for radar Scenario 1	69
6.6	ΔK per minibatch for radar Scenario 1	69
6.7	Aggregated performance metrics for radar Scenario 2	70
6.8	ΔK per minibatch for radar Scenario 2	70
6.9	Aggregated performance metrics for radar Scenario 3	71
6.10	ΔK per minibatch for radar Scenario 3	72
7.1	Clustered spike shapes for the spike sorting task	78
8.1	Pairwise feature plot of the heart disease dataset	90
8.2	Histograms for the categorical features in the heart disease dataset	90
8.3	Pairwise feature plot of the wine ratings dataset	95
8.4	Histograms for the categorical features in the wine ratings dataset	95

List of Tables

1.1	Publication list	5
1.2	Notation used in this document	7
3.1	Gaussian data model variables and parameters	24
5.1	Prior hyperparameters for synthetic data clustering tasks	47
5.2	Algorithm configurations for experiments with synthetic data	48
6.1	Emitter descriptions for the simulated scenario	63
6.2	Prior hyperparameters for radar pulse clustering	67
7.1	Prior hyperparameters for spike sorting	76
7.2	Algorithm performance on spike sorting	77
8.1	Presentation of example results for categorical variables	89
8.2	Features extracted from the heart disease dataset	89
8.3	Prior hyperparameters for heart disease clustering	91
8.4	Clusters found in the heart disease dataset with a mixed-data model and root-finding tuning method	93
8.5	Clusters found in the heart disease dataset with a mixed-data model and no tuning	93
8.6	Clusters found in the heart disease dataset with a fully Gaussian model and root-finding tuning	93
8.7	Encoding schemes used for categorical variables in the wine ratings dataset	94
8.8	Features extracted from the wine ratings dataset	96
8.9	Prior hyperparameters for wine ratings clustering	96
8.10	Clusters found in the wine ratings dataset with a mixed-data model and root-finding tuning method	97
8.11	Clusters found in the wine ratings dataset with a fully Gaussian model and root-finding tuning	98

Acknowledgment

First and foremost, thank you Ashley for your unwavering support along this journey. You have always been there for me. Thanks Mom, Dad, and Emily for your countless prayers and words of encouragement.

I owe my appreciation to a lot of people that I have met during my work on this dissertation—far too many to list here. To the following people, I want to say a special thank you: Jacob, for your close friendship, for suffering with me as a Ph.D. buddy, and for always being willing to talk about research; Tyler and Brittany, for being part of The Crew and for being two of my best friends since way back in undergrad; Doug, Andrew, Anne, and Tony, for supporting me at AFRL; David, for the many fruitful conversations and helpful feedback on my research; Vasu, for encouraging me to pursue this path of research at the very beginning; and Ryan, for helping me make the time to complete this work.

Thank you Dr. Garber, Dr. Shaw, Dr. Ash, and Dr. Gallagher for serving on my committee. Finally, thank you Dr. Rigling for the hours of mentoring, advising, and editing that you invested in me and this work. Had I never been your student, I may have never pursued any kind of research.

Thank you.

Dedicated to

Ashley

Chapter 1

Introduction

For decades, researchers have been developing new ways to extract meaning from unlabeled data using unsupervised learning and cluster analysis [1]–[3]. Data clustering has found applications in almost every scientific field. The vast extent of clustering applications has led to the development of a variety of algorithms, each with particular capabilities.

For example, some clustering algorithms are specialized for extracting clusters of a certain shape. These algorithms can be useful when the cluster shape is well-understood a priori, or when a particular cluster shape can provide special insights into the data. Example applications include astronomy [4], signal processing [5], [6], and biology [7]. Other algorithms place no restriction on the shape. This freedom is desirable when data clusters lie on arbitrary manifolds, which can occur in social networks [8] and x-ray imaging [9], among others.

Aside from the choice of cluster shape, several algorithms summarize data using a fixed number of clusters, which is appropriate when a user wants to know the general layout of the data in feature space [10]. When the number of clusters is not known a priori, the number of clusters needs to be inferred from the data. Applications for this capability include the environmental sciences [11], genetics [12], and image segmentation [13].

From a data processing perspective, many clustering algorithms operate in an offline

fashion and are intended to summarize the data for a user. For static data analysis like document classification [14], [15], this can be ideal. On the other hand, relatively few algorithms are designed to operate in an online or sequential fashion for streaming data—these methods can adapt dynamically to changes in the data and sometimes serve as pre-processing steps for downstream algorithms. Example applications of online clustering include network security [16] and radar signal processing [17]. Online clustering algorithms can also be used for large, static datasets by clustering subsets of the data and combining the results obtained from the subsets [18].

Finally, clustering algorithms may also be differentiated by the mathematical framework in which they are derived (we provide a brief taxonomy in Chapter 2). Model-based algorithms are derived in a probabilistic framework [19] and may be Bayesian (e.g. [20], [21]) or non-Bayesian (e.g. the maximum likelihood approach [22], [23]) in nature. Probabilistic methods usually compute a probability density over the cluster labels for each data point to represent the uncertainty in the assignment—this is often referred to as soft clustering [19]. Soft clustering can be useful when the boundaries between clusters are blurred and has found applications in marketing [24] and bioinformatics [25]. However, many model-based clustering methods require the number of clusters (the model order) to be specified a priori.

Bayesian nonparametric (BNP) methods can provide a way to sidestep this problem of model order selection in model-based clustering by utilizing specialized priors over the distribution of cluster assignments [26]. This can be a distinct advantage over other probabilistic model-based clustering algorithms like the Gaussian mixture model (GMM) and non-probabilistic algorithms like K -means [19], which require the number of clusters K to be specified a priori. In BNP clustering, the number of clusters is estimated as part of the inference process instead of being fixed. But BNP algorithms often over-estimate the number of clusters in the data [27], [28]—a phenomenon called over-clustering—leading researchers to develop cluster priors that specifically address this problem [29].

Inference in BNP models tends to be intractable, necessitating approximate inference frameworks such as Markov chain Monte Carlo (MCMC) or variational inference (VI) [26]. Although MCMC inference offers theoretical guarantees and MCMC samplers tend to have straightforward implementations, they suffer from slow convergence that is difficult to verify. VI techniques, on the other hand, tend to converge quickly and decisively, making them more appropriate for large datasets, but can be comparatively inaccurate due to local optima [30].

Some clustering applications, such as network intrusion detection [16] and radar signal processing [17], involve continuous streams of data whose underlying statistics may dynamically change over time. We use the term *online* clustering to describe such problems. Other works may use terms like streaming to refer to a similar paradigm, e.g., [31]–[33]. In these applications, VI is clearly preferable for model based clustering because of its convergence properties and natural extension to online updating [34]. When the number of clusters is unknown and/or not consistent throughout the dataset, BNP can be useful because it allows the number of clusters in the model to be flexible. Variational BNP clustering methods exist for offline [35], [36] and online [31] clustering, but do not explicitly address over-clustering. Recent work addressing the over-clustering problem relies on MCMC inference [29], which is inappropriate for online clustering. Improving the model order estimation in online BNP clustering remains an open problem to which we propose a solution.

1.1 Contributions

This dissertation bridges the gap between efficient online VI and BNP methods that are resistant to over-clustering. We contribute the following items to the body of clustering research:

1. **We derive an online variational BNP GMM with a powered Chinese Restaurant**

Process (PCRP) prior [29] to cluster dynamically changing datasets. Prior to this work, online BNP algorithms could be implemented using VI, but did not necessarily provide good estimates for the number of clusters in the data [27]. The MCMC-based method in [29] introduces the PCRP to address this, but it is not suitable for online clustering. Our algorithm advances the state of the art in BNP clustering by improving the estimation of the number of clusters in a dynamically changing dataset without relying on a priori knowledge.

2. **We propose two methods for tuning the PCRP hyperparameter in the online setting.** The PCRP features a tunable hyperparameter r that controls the tendency for new clusters to appear. Our novel methods for tuning this hyperparameter without supervision leverage concepts from Gaussian mixture reduction and an empirical study of the PCRP. In addition, we show that standard implementations of online BNP clustering algorithms can fail to adapt to changes in the cluster count over time. We incorporate recent advances in efficient change detection to address this issue and show that our hyperparameter tuning methods achieve superior results when compared to the standard approach.
3. **We demonstrate our algorithm on several datasets from diverse research areas.** We perform in-depth analysis for two synthetic numerical datasets with clusters that move dynamically through feature space and vary in count. Additionally, we show that our algorithm yields improved performance over standard online BNP clustering approaches in three radar pulse clustering simulations. Our algorithm also achieves competitive results on neural spike sorting tasks compared with specially-designed spike sorting algorithms, some of which do not operate in the online setting (making them inappropriate for use in vivo, for instance [37]).
4. **We extend our online BNP algorithm to cluster data with mixed Gaussian and Categorical types.** We demonstrate this model—the first of its kind—by clustering

Table 1.1: List of publications related to this dissertation. Status is current as of September 2020. All publications are authored by Matthew Scherreik and Brian Rigling.

Title	Venue	Reference	Status
Clustering radar pulses with Bayesian Nonparametrics: A case for online processing	2020 IEEE Radar Conference	[38]	Published
Controlling redundancy in on-line Bayesian nonparametric clustering	IEEE Transactions on Signal Processing	[39]	In review
Clustering radar pulses with online Bayesian nonparametrics for dynamic spectrum awareness	IEEE Transactions on Aerospace and Electronic Systems	[40]	In review

real-world datasets with a data mining perspective. We show that by leveraging the PCRP, our algorithm yields clusters with less redundancy. Online processing enables us to cluster a very large dataset in a short amount of time.

1.2 Published and submitted papers related to this dissertation

Portions of this dissertation have been published in a refereed conference and submitted to academic journals for peer review. Our conference publication shows that an online variational implementation of a Bayesian nonparametric model is more appropriate for clustering radar pulses than MCMC [38]. We present a new algorithm for clustering data with online BNP in [39] along with a novel hyperparameter tuning method. Finally, we extend the simulation and results we first presented in [38] to include our newly developed algorithm and tuning method in [40]. Table 1.1 lists our publication titles, venues, and status.

1.3 Outline

The remainder of this dissertation is organized as follows. Theoretical background and related work are reviewed in Chapter 2. We derive a novel variational inference algorithm for clustering data with a Bayesian nonparametric Gaussian mixture model in Chapter 3. Our algorithm relies on a critical hyperparameter in order to achieve good results—we present two new methods for tuning this hyperparameter in an online fashion in Chapter 4. Chapters 5 to 7 present empirical studies of our new GMM in three application areas. The first involves dynamically changing synthetic data that enables in-depth analysis of the algorithm’s performance. In the second set of experiments (Chapter 6), we apply the algorithm to the problem of radar pulse clustering, a relevant problem in cognitive radio. The third set of experiments (Chapter 7) involves neural spike sorting, a process by which electrical impulses detected in the brain are clustered into similar shapes. We show that our model-based framework is not limited to Gaussian data by extending our approach to data of mixed type in Chapter 8. This involves a revised probabilistic model and newly extended hyperparameter tuning methods. Finally, we make concluding remarks and provide avenues for continued research in Chapter 9.

1.4 Notation

See Table 1.2 for a list of our notation. Sets are denoted by calligraphic letters such as $\mathcal{X}$. Vectors and matrices are denoted by boldface lowercase and uppercase letters like $\mathbf{x}$ and $\mathbf{X}$, respectively. Vertical bars ($|\cdot|$) indicate the absolute value or matrix determinant when the argument is a scalar or matrix, respectively. Scalar variables are represented by lowercase italic letters as in z . Quantities are denoted by capital italic letters like N . For minibatch indices when discussing online processing, we use a superscript like $\mathcal{X}^t$. Other local notation is explained throughout the text.

Table 1.2: Notation used in this document

Description	Example
Data point n	$\mathbf{x}_n \in \mathcal{R}^D$
Data dimensionality	D
Cluster assignment for $\mathbf{x}_n$	$z_n \in \mathcal{N}$
Cluster parameters for cluster k	$\boldsymbol{\theta}_k$
Set of data points	$\mathcal{X}$
Set of cluster assignments	$\mathcal{Z}$
Set of cluster parameters	Θ
Item with element n removed	$N_{k,\eta}, \mathcal{Z}_{\eta}$

Chapter 2

Background

In this chapter we provide technical background information that is foundational for the rest of the dissertation. We start with a review of the clustering literature in which we present a brief taxonomy of clustering algorithms, discuss estimation of the number of clusters (or lack thereof), differentiate offline and online clustering, and review related work found elsewhere in the literature. We then provide an in-depth discussion of variational inference for BNP clustering, upon which the remainder of the dissertation is based. Finally, we show how variational inference is utilized for online clustering.

2.1 Literature review

Clustering is a data analysis method by which data is sorted into categories (clusters) [41]. Data points in the same cluster are similar to one another in some way, while points in separate clusters are somehow dissimilar. It is considered an unsupervised learning technique—in contrast with supervised learning (i.e., classification), ground truth label information is not available.

The lack of ground truth complicates objective assessment of clustering algorithm

outputs. Imposing the notion of a “correct” clustering solution essentially provides supervision. The subjectivity of the clustering problem has led to the development of a variety of approaches. While most clustering algorithms have the same goal—to somehow make sense of unlabeled data—they each impose different assumptions that impact the final result. In this review, we first provide a brief taxonomy of clustering algorithms in general. For brevity, we limit this discussion to the most basic concepts and refer to the reader to one of many surveys on data clustering for more details [42]–[44]. We then review clustering topics that pertain to our application and algorithm development, namely, choosing the number of clusters in the data and online clustering.

2.1.1 Brief taxonomy of clustering algorithms

Despite the large number of clustering techniques, most algorithms fall into one of four categories:

1. **Centroid-based.** Each cluster is represented by a vector center. Data is clustered by arranging K cluster centers such that they meet some goodness criteria. Most often, the cluster center placement minimizes within-cluster variation and maximizes inter-cluster distance. Centroid-based techniques require the number of clusters K to be fixed a priori. Common distance metrics include Euclidean and ℓ_1 distance. Centroid-based clustering algorithms assume that the clusters in the data are convex—this means that clusters do not have arbitrary shape. *Example algorithms:* K -means [45], K -medoids [46].
2. **Hierarchical.** Data is organized into a tree structure based on the distance between points and the distance between clusters. The distance measure is arbitrary, and the tree may be built from the bottom-up (agglomerative) or from the top-down (divisive). A user selects the number of clusters by choosing a level of the tree to “cut.” Hierarchical methods can find arbitrarily-shaped clusters in the data. *Example algo-*

gorithms: BIRCH [47], CURE [48], Chameleon [49].

3. **Density-based.** Clusters are formed in areas of feature space with high data density by constructing a graph between data points. Data points are connected by comparing the distance between them to a chosen threshold. Unlike hierarchical methods, however, not all data points are connected, so a complete tree structure is not formed. Clusters can have arbitrary shape and the number of clusters does not need to be specified a priori—the number of clusters is a byproduct of algorithm parameters. *Example algorithms: DBSCAN [50], OPTICS [51], SOC [52].*

4. **Model-based.** Clusters are assumed to take a form specified by a probabilistic model. The most common model is a mixture of Gaussian distributions. Typically, the number of clusters is specified a priori; however, extensions exist to estimate it from data (we discuss this in more detail later). In contrast with other clustering methods which provide hard cluster labels, model-based methods often estimate a soft assignment—a probability distribution over cluster labels—for each data point. In general, inference in model-based clustering is computationally expensive, but this can be addressed by way of approximation. *Example algorithms: GMM [53], BNP/IGM [54], [55].*

The classes of clustering algorithms described here encompass the majority of techniques in use today. Others, such as ones based on graph theory or fuzzy set theory, can either be seen as subsets of the ones listed here or as fringe methods without much activity in the literature.

2.1.2 Number of clusters

Specifying or otherwise determining the correct number of clusters in the data is a central problem in all cluster analysis. Many clustering algorithms including K -means and GMM require that the number of clusters be specified a priori. This parameter is held fixed during

the clustering process. However, the number of clusters in the data is often unknown in advance. In fact, obtaining this information may be a desired result of cluster analysis in the first place, which has led to the development of methods to determine the number of clusters algorithmically [56]–[58]. These methods generally operate independently of the clustering algorithm, so they can be paired with any approach. Many external methods require the algorithm to be run multiple times on the same dataset while varying the pre-specified number of clusters in order to determine the best solution [59].

Alternatively, a class of techniques called Bayesian nonparametrics [60] sidesteps this issue by placing a specialized prior distribution on the cluster assignments. The prior implies that there is an infinite number of clusters for an infinitely large dataset—but any finite number of data points only realizes a finite number of clusters. The number of clusters is thus data-driven and found as a byproduct of the inference process and hyperparameter tuning [54], [55]. Classical BNP models are known to over-estimate the number of clusters in the data even with optimal hyperparameter settings [27]. We further discuss this tendency in Section 3.1.

2.1.3 Offline vs. online clustering

Most of the clustering algorithms in the literature today are designed for offline data processing. In other words, they operate on one large batch of data and often require multiple iterations over the dataset before reaching a clustering solution. This contrasts with the online (or streaming/sequential) clustering paradigm, where data is assumed to arrive in a continuous stream, thereby limiting (or prohibiting) access to the entire dataset. Online algorithms can also operate on large, static datasets by splitting the dataset into subsets and computing an overall solution by combining results obtained from the subsets.

Online extensions of offline clustering algorithms have been studied in the literature. For example, numerous online K -means clustering algorithms have been developed over the years [32], [33], [61], [62]. Some algorithms, such as the density-based SOC, are not

online extensions of existing offline algorithms, but are derived with streaming applications in mind from the start [52]. Online versions of model-based clustering algorithms are also widespread in the literature [63]–[65]. Recent activity in online clustering research indicates that it is an open problem with ample room for new developments.

2.1.4 Related work

We now focus on advances in BNP model-based clustering, because these techniques are instrumental in our proposed approach. As mentioned previously, the BNP framework enables the user to avoid the problem of specifying the number of clusters in the data before performing cluster analysis. A major drawback of these approaches, however, is the general intractability of the models, necessitating an approximate inference approach such as Markov chain Monte Carlo (MCMC) [66] or variational inference (VI) [30].

Although there is ongoing work to speed up MCMC inference in general [67]–[69], VI has traditionally been the framework of choice for BNP clustering when speed and efficiency are desired [34]–[36]. As a result, VI has served as a basis for the development of online BNP clustering algorithms [31] and other streaming applications of BNP within machine learning [70], [71]. We describe VI for BNP clustering in more detail in Section 2.2.

Aside from the issue of computational complexity, it is well-known that BNP models can over-estimate the number of clusters in the data, especially when the dataset is large [27]. This phenomenon is sometimes referred to as over-clustering. It can be shown that for classical BNP clustering models, the expected number of clusters grows logarithmically with the data [72]. BNP models are equipped with a hyperparameter to control the influence of the data count, which can itself be inferred due to the Bayesian framework [73]. However, even with careful tuning, the over-clustering phenomenon is difficult to manage, leading researchers to develop new BNP priors with additional hyperparameters [29].

2.2 Variational BNP clustering

Model-based clustering begins with a generative model that specifies the relationships between the data $\mathcal{X}$, the hidden variables (cluster assignments) $\mathcal{Z}$, and cluster parameters Θ . A typical clustering model is given by [36]

$$p(\mathcal{X}, \mathcal{Z}, \Theta) = p(\mathcal{X} \mid \mathcal{Z}, \Theta)p(\mathcal{Z})p(\Theta) \quad (2.1)$$

$$= \left[\prod_{n=1}^N p(\mathbf{x}_n \mid z_n, \boldsymbol{\theta}_k) \right] p(\mathcal{Z}) \left[\prod_{k=1}^{\infty} p(\boldsymbol{\theta}_k) \right], \quad (2.2)$$

which is often referred to as a “collapsed” model, because there is no term to explicitly model mixture weights—they have been integrated out. For BNP, $p(\mathcal{Z})$ represents a specialized prior over the cluster assignments. Inferring the posterior $p(\mathcal{Z}, \Theta \mid \mathcal{X})$ in these models is intractable due to the exponential complexity of computing the evidence, so approximations are necessary [35]. As mentioned previously, inference is often performed via MCMC [74], which is not appropriate for clustering large datasets or for online clustering [30].

Variational inference (VI) is an alternative framework which casts inference as an optimization problem [19]. A simplified “mean-field” model

$$q(\mathcal{Z}, \Theta) = q(\mathcal{Z})q(\Theta) \quad (2.3)$$

$$= \prod_{n=1}^N q(z_n) \prod_{k=1}^{\infty} q(\boldsymbol{\theta}_k). \quad (2.4)$$

is proposed and the Kullback-Leibler divergence $\text{KL}(q \parallel p)$ is minimized by equivalently

maximizing the evidence lower bound

$$\text{ELBO}(q) = \mathbb{E} [\log p(\mathcal{X}, \mathcal{Z}, \Theta)] - \mathbb{E} [\log q(\mathcal{Z}, \Theta)] \quad (2.5)$$

$$\begin{aligned} &= \mathbb{E} \left[\sum_{n=1}^N \log p(\mathbf{x}_n \mid z_n, \boldsymbol{\theta}_{z_n}) \right] + \mathbb{E} [\log p(\mathcal{Z})] + \mathbb{E} \left[\sum_{k=1}^{\infty} \log p(\boldsymbol{\theta}_k) \right] \\ &\quad - \mathbb{E} \left[\sum_{n=1}^N \log q(z_n) \right] - \mathbb{E} \left[\sum_{k=1}^{\infty} \log q(\boldsymbol{\theta}_k) \right], \end{aligned} \quad (2.6)$$

where the expectations are taken with respect to $q(\mathcal{Z}, \Theta)$ [36]. Note that although there are sums in (2.2), (2.4) and (2.6) with an infinite number of terms, most of these terms are zero in practice. These sums are only computed over the number of clusters that contain data—but the BNP framework allows this number to change during inference.

Coordinate ascent is typically employed to optimize the ELBO [30]. The parameters θ_k and hidden variables z_n are updated individually while holding the rest of the variables constant. More specifically, the coordinate ascent update equations for the cluster parameters and cluster assignments are given by [19]

$$\log q^*(\boldsymbol{\theta}_k) = \sum_{n=1}^N \mathbb{E} [\log p(\mathbf{x}_n \mid \boldsymbol{\theta}_k)] + \log p(\boldsymbol{\theta}_k) + \text{CONST} \quad (2.7)$$

$$\begin{aligned} \log q^*(z_n) &= \mathbb{E} [\log p(\mathbf{x}_n \mid \boldsymbol{\theta}_{z_n})] + \mathbb{E} [\log p(z_n \mid \mathcal{Z}_{\setminus n})] \\ &\quad + \text{CONST}, \end{aligned} \quad (2.8)$$

where the expectations are taken with respect to the variational density q for all of the other variables, i.e., $q(\mathcal{Z}, \Theta_{\setminus k})$ for (2.7) and $q(\mathcal{Z}_{\setminus n}, \Theta)$ for (2.8). The ELBO is optimized by iterating through the update equations in a manner similar to the expectation-maximization method, leading to this algorithm being known as “variational EM” [75], [76]. The result is a variational approximation to the desired posterior $q^*(\mathcal{Z}, \Theta) \approx p(\mathcal{Z}, \Theta \mid \mathcal{X})$.

Implementation of the updates (2.7) and (2.8) is simplified when the data model $p(\mathcal{X} \mid$

$\mathcal{Z}, \Theta$) is in the conjugate exponential family [75]. For example, components in a mixture of multivariate Normal distributions with unknown mean and unknown covariance have a Normal Wishart conjugate prior, which enables us to compute (2.7) and (2.8) in closed form [19]. We employ this data model throughout the majority of this work. Computations corresponding to this model are provided in Section 3.3.

A BNP prior can be incorporated via the $E[\log p(z_n \mid \mathcal{Z}_{\mathcal{U}})]$ term in (2.8), enabling the number of clusters to be inferred from the data as part of the VI optimization process. The Chinese restaurant process (CRP) [77] is a common choice. The CRP can be described by the following analogy. Imagine a restaurant with an infinite number of tables. Customer $n = 1$ enters the restaurant and sits at table $k = 1$. The next customer, numbered $n = 2$, must choose whether to sit at the already occupied table $k = 1$ or sit at a new unoccupied table $k = 2$. The probabilities associated with each choice are $1/\alpha + 1$ and $\alpha/\alpha + 1$, respectively, for some fixed $\alpha > 0$. As more customers $n = 3, 4, \dots, N$ enter the restaurant, the assignment z_n of the n -th customer to table k has probability [77]

$$p(z_n = k \mid \mathcal{Z}_{\mathcal{U}}) = \begin{cases} \frac{N_{k,\mathcal{U}}}{\alpha + N - 1} & k = 1, \dots, K \\ \frac{\alpha}{\alpha + N - 1} & k > K, \end{cases} \quad (2.9)$$

where K is the number of occupied tables at that time, $\mathcal{Z}$ is the set of table assignments, and $\mathcal{Z}_{\mathcal{U}}$ and $N_{k,\mathcal{U}}$ are the set of table assignments and counts at each table without counting customer n , respectively. Substituting data points and clusters for customers and tables, respectively, we obtain a model for a process by which data points are assigned cluster labels. We discuss another choice of BNP prior in Section 3.1.

It was noted in [36], [78] that direct implementation of the CRP into (2.8) is intractable for even moderately large data sizes N . Examining the expectation

$$E[\log p(z_n \mid \mathcal{Z}_{\mathcal{U}})] = \sum_{\mathcal{Z}_{\mathcal{U}}} \prod_{m \neq n} q(z_m) \log p(z_n \mid \mathcal{Z}_{\mathcal{U}}) \quad (2.10)$$

reveals a sum over $K^{(N-1)}$ possible arrangements of assignments to K clusters. A tractable approximation of the expectation can be obtained by exploiting the fact that $p(z_n \mid \mathcal{Z}_{\mathcal{U}})$ is specified in terms of counts of data points in a cluster. Since these counts can be expressed as sums of indicator functions, i.e., $N_k = \sum_{n=1}^N \mathbb{1}[z_n = k]$, and the assignment z_n to cluster k has probability $q(z_n = k)$, they can be treated as sums of Bernoulli random variables. When N is large enough, N_k is well-approximated by a Gaussian random variable with mean and variance given by [36], [78]:

$$\mathbb{E}[N_k] \approx \sum_{n=1}^N q(z_n = k) \quad (2.11)$$

$$\mathbb{V}[N_k] \approx \sum_{n=1}^N q(z_n = k)(1 - q(z_n = k)), \quad (2.12)$$

where $q(z_n = k)$ the value of the variational distribution for the cluster assignment $z_n = k$. Then a second-order Taylor approximation, i.e. [79]

$$\mathbb{E}[f(x)] \approx f(\mathbb{E}[x]) + f''(\mathbb{E}[x]) \mathbb{V}[x]/2, \quad (2.13)$$

can be used to approximate $\mathbb{E}[\log p(z_n \mid \mathcal{Z}_{\mathcal{U}})]$ as shown in [36], [78].

2.2.1 Online variational BNP clustering

Inference in the VI framework is often much faster than MCMC. Recent work has leveraged the efficiency of VI for online BNP clustering, where data points arrive in a continuous stream [31]. In this processing paradigm, data is collected into a minibatch (buffer) before being clustered. Minibatches are processed sequentially in the following way. Let a minibatch $\mathcal{X}^t = \{\mathbf{x}_n\}_{n=1}^{N^t}$ be indexed by $t = 1, 2, \dots$. Furthermore, to simplify the notation, let all hidden variables and parameters be grouped together in the variable Θ . Following [34],

applying Bayes' rule to the first minibatch yields

$$p(\Theta \mid \mathcal{X}^1) = \frac{p(\mathcal{X}^1 \mid \Theta)p(\Theta)}{p(\mathcal{X}^1)}. \quad (2.14)$$

Bayes' rule can be applied to the following minibatches $t > 1$ using the posterior from all $t - 1$ previous minibatches as a replacement for the original prior $p(\Theta)$, yielding the update [34]

$$p(\Theta \mid \mathcal{X}^{1:t}) = \frac{p(\mathcal{X}^t \mid \Theta)p(\Theta \mid \mathcal{X}^{1:t-1})}{p(\mathcal{X}^{1:t-1})}, \quad (2.15)$$

where the notation $\mathcal{X}^{1:t}$ means we are conditioning on minibatches 1 through t .

Recall that VI approximates the posterior $p(\Theta \mid \mathcal{X})$ by a factorized density $q(\Theta)$. Thus, an online variational inference algorithm can be obtained by substituting the true posterior with its variational approximation [34]:

$$p(\Theta \mid \mathcal{X}^{1:t}) \approx q^t(\Theta) \quad (2.16)$$

$$= \frac{p(\mathcal{X}^t \mid \Theta)q^{t-1}(\Theta)}{p(\mathcal{X}^{1:t-1})}, \quad (2.17)$$

where $q^t(\Theta)$ is the variational posterior for minibatch t . From this discussion it is clear that VI is a natural choice for online processing. We rely on the tractability of computing the variational posterior to streamline the posterior update.

2.3 Summary

In this chapter, we reviewed the clustering literature and provided prerequisite technical material for the remainder of this dissertation. We provided a succinct taxonomy of clustering algorithms, reviewed methods for estimating the number of clusters, and discussed related work found elsewhere in the literature. We then derived the variational BNP clus-

tering framework which the rest of this dissertation builds upon. This includes the ELBO (2.5), coordinate ascent updates (2.7) and (2.8), and motivation for a Taylor approximation of $E[\log p(z_n \mid \mathcal{Z}_n)]$. We also showed how variational inference can be used for online clustering.

Chapter 3

Algorithm derivation

Although the BNP framework sidesteps the issue of choosing the number of clusters a priori, it is known that the inferred number of clusters tends to be too high [27]. This is sometimes called “over-clustering.” Recent work addressing the over-clustering problem relies on MCMC inference [29], which is inappropriate for online clustering. In this chapter, we employ variational inference to derive a new online BNP clustering algorithm that is resistant to over-clustering.

3.1 Redundancy in BNP clustering

Inspection of (2.9) reveals two key attributes of the CRP. First, the CRP exhibits a “rich-get-richer” property—that is, large clusters tend to attract more data points. Second, as amount of data increases, the number of clusters is also expected to increase. In fact, it has been shown that the expected number of clusters grows logarithmically with N [72].

Although the CRP exhibits nice theoretical properties such as infinite exchangeability (i.e., the order of the data does not impact the probability of a partition), BNP clustering algorithms that utilize this prior have practical issues. In particular, it is well-known that

these algorithms tend to create too many clusters when the number of data points is large—a phenomenon that is sometimes called “over-clustering” [27], [28]. It is possible to tune the parameter α to prevent this tendency, but it is not always effective [27], [29]. Alternatively, [29] showed that the CRP can be modified to allow for greater flexibility in modeling the way that cluster counts N_k impact the probability of a partition by raising the cluster counts to a power $r > 1$, i.e.,

$$p(z_n = k \mid \mathcal{Z}_{\mathcal{U}}) = \begin{cases} \frac{N_{k,\mathcal{U}}^r}{\alpha + \sum_{\ell=1}^K N_{\ell,\mathcal{U}}^r} & k = 1, \dots, K \\ \frac{\alpha}{\alpha + \sum_{\ell=1}^K N_{\ell,\mathcal{U}}^r} & k > K, \end{cases} \quad (3.1)$$

called the powered CRP or PCR. ¹ It was shown in [29] that this prior affords greater control over the number of clusters in the final mixture when compared to the concentration parameter α , so we utilize it in our work. However, their MCMC implementation was intended only for offline use.

It is desirable for a clustering algorithm to be able to estimate the number of clusters in the data automatically, without creating redundant clusters. This is especially important when dealing with streaming data that can have a dynamically changing number of clusters. The PCR prior can reduce over-clustering, but lacks an online implementation. We use the VI framework to derive a novel online BNP clustering algorithm with the PCR prior.

3.2 Reducing redundancy in online BNP clustering

We discussed the general VI framework for BNP clustering in Section 2.2. To implement the PCR (3.1) in the variational update equation (2.8), we apply (2.13) to obtain a Taylor approximation of $E[\log p(z_n \mid \mathcal{Z}_{\mathcal{U}})]$ like in [36], [78]. We first present the final approximation, then follow with a full derivation. To simplify the notation, let

¹Note that setting $r = 1$ in (3.1) recovers the standard CRP as given in (2.9).

$\mu_{k,\mathcal{Y}} \triangleq \mathbb{E}[N_{k,\mathcal{Y}}]$ and $\sigma_{k,\mathcal{Y}}^2 \triangleq \mathbb{V}[N_{k,\mathcal{Y}}]$ as defined in (2.11) and (2.12), respectively. Furthermore let $D \triangleq \alpha + \sum_{k=1}^K \mu_{k,\mathcal{Y}}^r$. The Taylor approximation of the expected log-PCR is given by

$$\mathbb{E}[\log p(z_n = k \mid \mathcal{Z}_{\mathcal{Y}})] \approx \begin{cases} \log \frac{\mu_{k,\mathcal{Y}}^r}{D} + \frac{\sigma_{k,\mathcal{Y}}^2}{2} \left[\frac{(r\mu_{k,\mathcal{Y}}^{r-1})^2}{D^2} + \frac{(r-r^2)\mu_{k,\mathcal{Y}}^{r-2}}{D} - \frac{r}{\mu_{k,\mathcal{Y}}^2} \right] & k \leq K \\ \log \frac{\alpha}{D} & k > K. \end{cases} \quad (3.2)$$

To derive this, first let the denominators in (3.1) be re-stated as

$$D = \beta_k + \mu_{k,\mathcal{Y}}^r, \quad (3.3)$$

where $\beta_k = \alpha + \sum_{\ell \neq k} \mu_{\ell,\mathcal{Y}}^r$ isolates the term $\mu_{k,\mathcal{Y}}^r$. Thus for the $k = 1, \dots, K$ term in (3.1), the function in the Taylor approximation (2.13) is $f(\mu_{k,\mathcal{Y}}) = g(h(\mu_{k,\mathcal{Y}}))$, a composite function with $h(\mu_{k,\mathcal{Y}}) = \mu_{k,\mathcal{Y}}^r$ and $g(\mu_{k,\mathcal{Y}}) = \log \mu_{k,\mathcal{Y}} - \log(\beta_k + \mu_{k,\mathcal{Y}}^r)$. Inserting these definitions into (2.13) yields

$$\mathbb{E}[g(h(N_{k,\mathcal{Y}}))] \approx g(h(\mu_{k,\mathcal{Y}})) + \frac{\sigma_{k,\mathcal{Y}}^2}{2} g''(h(\mu_{k,\mathcal{Y}})) \quad (3.4)$$

$$= g(h(\mu_{k,\mathcal{Y}})) + \frac{\sigma_{k,\mathcal{Y}}^2}{2} [h'(\mu_{k,\mathcal{Y}})^2 g''(h(\mu_{k,\mathcal{Y}})) + h''(\mu_{k,\mathcal{Y}}) g'(h(\mu_{k,\mathcal{Y}}))] , \quad (3.5)$$

which results from applying the chain and product rules to $g''(h(\mu_{k,\mathcal{Y}}))$. Plugging the derivatives

$$h'(\mu_{k,\mathcal{Y}}) = r\mu_{k,\mathcal{Y}}^{r-1} \quad (3.6)$$

$$h''(\mu_{k,\mathcal{Y}}) = (r^2 - r)\mu_{k,\mathcal{Y}}^{r-2} \quad (3.7)$$

$$g'(\mu_{k,\mathcal{Y}}) = \mu_{k,\mathcal{Y}}^{-1} - (\beta_k + \mu_{k,\mathcal{Y}}^r)^{-1} \quad (3.8)$$

$$g''(\mu_{k,\mathcal{Y}}) = (\beta_k + \mu_{k,\mathcal{Y}}^r)^{-2} - \mu_{k,\mathcal{Y}}^{-2}. \quad (3.9)$$

into (3.5) and rearranging terms yields the desired Taylor approximation

$$\mathbb{E}[\log p(z_n = k \mid \mathcal{Z}_{\mathcal{I}})] \approx \log \frac{\mu_{k,\mathcal{I}}^r}{D} + \frac{\sigma_{k,\mathcal{I}}^2}{2} \left[\frac{(r\mu_{k,\mathcal{I}}^{r-1})^2}{D^2} + \frac{(r-r^2)\mu_{k,\mathcal{I}}^{r-2}}{D} - \frac{r}{\mu_{k,\mathcal{I}}^2} \right] \quad (3.10)$$

for $k = 1, \dots, K$. Noting that the $k > K$ term in (3.1) is constant with respect to $N_{k,\mathcal{I}}$, its derivatives are zero. We thus arrive at the second-order Taylor approximation of the expected log-PCR as given in (3.2).

3.3 Inference for multivariate Gaussian data

We now provide details regarding our implementation, using the derivations in Section 3.2. In this work, we implement a collapsed variational BNP GMM according to the following generative model: For brevity, we will omit the terms “collapsed” and “multivariate” when there is no confusion. This differs from a traditional GMM in several ways (see [19] for a good summary of the standard and Bayesian GMM). First, the traditional GMM is computed via the expectation maximization (EM) algorithm for maximum likelihood estimation (MLE), resulting in hard cluster assignments. Our implementation, on the other hand, uses variational EM for a fully Bayesian solution, i.e., we will specify priors on the model parameters and compute their distributions instead of the MLE point estimate. Second, the typical GMM explicitly parameterizes mixture weights. We use a collapsed model, where the weights are integrated out in order to reduce the number of parameters in the model and allow us to explicitly model $p(z_n \mid \mathcal{Z}_{\mathcal{I}})$. Finally, traditional and Bayesian GMMs simply model the cluster labels as draws from a categorical distribution. Our approach uses BNP priors (the PCR in particular) to model the cluster labels.

We first state our generative model, then provide the equations for variational EM under this model. Let the n -th D -dimensional data point be represented by $\mathbf{x}_n \in \mathcal{R}^D$. Our

generative model is based on other work involving the PCRCP [29]:

$$\boldsymbol{\theta}_k = \boldsymbol{\mu}_k, \mathbf{T}_k \quad (3.11)$$

$$\mathbf{x}_n \mid \boldsymbol{\mu}_k, \mathbf{T}_k \stackrel{\text{iid}}{\sim} \text{Norm}(\mathbf{x}_n; \boldsymbol{\mu}_k, \mathbf{T}_k^{-1}) \quad (3.12)$$

$$\boldsymbol{\mu}_k, \mathbf{T}_k \mid \mathbf{m}_0, \nu_0, \mathbf{W}_0, \lambda_0 \sim \text{NormWish}(\boldsymbol{\mu}_k, \mathbf{T}_k; \mathbf{m}_0, \nu_0, \mathbf{W}_0, \lambda_0) \quad (3.13)$$

$$z_n \mid \mathcal{Z}_n, \alpha, r \sim \text{PCRCP}(z_n; \mathcal{Z}, \alpha, r), \quad (3.14)$$

where

$$\text{NormWish}(\boldsymbol{\mu}_k, \mathbf{T}_k; \mathbf{m}_0, \nu_0, \mathbf{W}_0, \lambda_0) = \text{Norm}(\boldsymbol{\mu}_k; \mathbf{m}_0, \nu_0 \mathbf{T}_k) \text{Wish}(\mathbf{T}_k; \mathbf{W}_0, \lambda_0) \quad (3.15)$$

with the Normal density function parameterized by the precision matrix $\mathbf{T}$ given by [19]

$$\text{Norm}(\mathbf{x}; \boldsymbol{\mu}, \mathbf{T}) = (2\pi)^{-\frac{D}{2}} |\mathbf{T}|^{\frac{1}{2}} \exp \left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^\top \mathbf{T} (\mathbf{x} - \boldsymbol{\mu}) \right) \quad (3.16)$$

and the Wishart density function given by [19]

$$\text{Wish}(\mathbf{T}; \mathbf{W}, \lambda) = \frac{|\mathbf{T}|^{\frac{\lambda-D-1}{2}} \exp \left(-\frac{1}{2} \text{tr}(\mathbf{W}^{-1} \mathbf{T}) \right)}{2^{\frac{\lambda D}{2}} |\mathbf{W}|^{\frac{\lambda}{2}} \pi^{\frac{D(D-1)}{4}} \prod_{i=1}^D \Gamma \left(\frac{\lambda+1-i}{2} \right)}. \quad (3.17)$$

The gamma function is denoted $\Gamma(\cdot)$ [19]. The model described in (3.11) to (3.14) also appears in [29], but they derive an offline MCMC algorithm to infer the parameters. We use variational inference with our approximation (3.2) for a novel online implementation. Other similar data models appear elsewhere [31], [35], [36], but they do not utilize the PCRCP to resist over-clustering. Table 3.1 lists the model variables, parameters, and dimensions.

The variational E-step entails updating the cluster assignment probabilities via the update equation for $q(z_n)$ given in (2.8). The update has two terms: the expected log

Table 3.1: Variables and parameters used for our Gaussian data model. The number of data points is denoted N .

Name	Description	Dimension
$\mathbf{x}_n$	n -th data point	D
$\boldsymbol{\mu}_k$	Mean of the k -th cluster	D
$\mathbf{T}_k$	Precision matrix for the k -th cluster	$D \times D$
$\mathbf{m}_0, \nu_0, \mathbf{W}_0, \lambda_0$	Normal-Wishart prior hyperparameters	$D, 1, D \times D, 1$
z_n	Cluster assignment of the n -th data point	1
$\mathcal{Z}_{\eta'}$	Cluster assignments of all data points except n	$N - 1$
α	CRP concentration hyperparameter	1
r	PCR regularization exponent hyperparameter	1

likelihood under the variational distribution, and the expected log PCR. The first term can be written as

$$\mathbb{E}[\log p(\mathbf{x} \mid \boldsymbol{\mu}_k, \mathbf{T}_k)] = \frac{1}{2} \left[-D \log(2\pi) + \mathbb{E}[\log |\mathbf{T}_k|] - \mathbb{E}[(\mathbf{x}_n - \boldsymbol{\mu}_k)^\top \mathbf{T}_k (\mathbf{x}_n - \boldsymbol{\mu}_k)] \right], \quad (3.18)$$

with the standard results [19]

$$\mathbb{E}[\log |\mathbf{T}_k|] = \sum_{i=1}^D \psi \left(\frac{\lambda_k + 1 - i}{2} \right) + D \log 2 + \log |\mathbf{W}_k| \quad (3.19)$$

$$\mathbb{E}[(\mathbf{x}_n - \boldsymbol{\mu}_k)^\top \mathbf{T}_k (\mathbf{x}_n - \boldsymbol{\mu}_k)] = D\nu_k^{-1} + \lambda_k (\mathbf{x}_n - \mathbf{m}_k)^\top \mathbf{W}_k (\mathbf{x}_n - \mathbf{m}_k), \quad (3.20)$$

where ψ is the digamma function. The values of $\mathbf{m}_k$, ν_k , $\mathbf{W}_k$, and λ_k are determined in the variational M-step. The second term in the update (2.8) is the PCR, given in equation (3.2).

The variational M-step is a standard Bayes hyperparameter update [19]:

$$\nu_k = \nu_0 + M_k \quad (3.21)$$

$$\boldsymbol{\mu}_k = \nu_k^{-1}(\nu_0 \mathbf{m}_0 + M_k \bar{\mathbf{x}}_k) \quad (3.22)$$

$$\mathbf{W}_k^{-1} = \mathbf{W}_0^{-1} + M_k \mathbf{S}_k + \frac{\nu_0 M_k}{\nu_0 + M_k} (\bar{\mathbf{x}}_k - \mathbf{m}_0)(\bar{\mathbf{x}}_k - \mathbf{m}_0)^\top \quad (3.23)$$

$$\lambda_k = \lambda_0 + M_k + 1 \quad (3.24)$$

However, since the expectations are taken with respect to variational distribution, the sample statistics are computed as follows [19]:

$$M_k = \sum_{n=1}^N q(z_n = k) \quad (3.25)$$

$$\bar{\mathbf{x}}_k = \frac{1}{M_k} \sum_{n=1}^N q(z_n = k) \mathbf{x}_n \quad (3.26)$$

$$\mathbf{S}_k = \frac{1}{M_k} \sum_{n=1}^N q(z_n = k) (\mathbf{x}_n - \bar{\mathbf{x}}_k)(\mathbf{x}_n - \bar{\mathbf{x}}_k)^\top. \quad (3.27)$$

3.4 Remarks on implementation

It is important to note that VI using a coordinate ascent strategy (as we use here) only guarantees convergence to a local optimum [30]. This means that we need to be careful about initializing the algorithm properly [80]. VI initialization is an ongoing research topic [81], [82]. We require an initial number of clusters K_0 to be specified, but this value does not fix the number of clusters in the model—it is only an initial guess. One initialization strategy is to compute a random variational distribution $q(z_n)$ for each data point $\mathbf{x}_n$. This can be implemented by drawing K_0 values from $\text{Unif}(0, 1)$ for each z_n and normalizing each $q(z_n)$ to sum to unity. Alternatively, one can obtain an initial clustering solution using K -means for $K = K_0$ clusters [83]. K -means only provides hard cluster assignments, meaning that

$q(z_n = k^*) = 1$ for the assigned cluster k^* and $q(z_n = k) = 0$ for all $k \neq k^*$. Because we operate on soft cluster assignments, we “soften” the K -means estimate by adding a small ϵ to all $q(z_n)$ and re-normalizing each $q(z_n)$ to sum to one. K -means initialization may be the most popular initialization strategy for VI clustering and is implemented by the `scikit-learn` package [84]. We have found that initializing using K -means generally provides a better starting point for VI without significantly adding to the computational complexity.

Prior Bayesian clustering work has established that VI methods tend to perform better when K_0 is large relative to the true number of clusters in the data [85], [86]. If K_0 is too small, the algorithm may increase the covariance of an existing cluster rather than create a new one. In other words, unnecessary clusters are easily pruned from the model, but new clusters are less likely to be added during coordinate ascent. Empirical analysis indicates that this notion holds true for our algorithm as well. As such, we advise setting K_0 large and allowing the inference process to delete unnecessary clusters.

Despite this tendency, it is important to describe the mechanism by which clusters can be added to the model automatically. We compute $q(z_n)$ for $K + 1$ clusters—the first K are “true clusters” that are actually realized by the data. The remaining cluster is an unrealized cluster. If, at the end of an iteration, $N_{K+1} > 0$ (implying that there are points which do not currently fit into one of the K realized clusters), the number of clusters K is incremented by one and the corresponding points are added to the new cluster. At the next iteration, the parameters of the new cluster will be estimated via (2.7). The algorithm also checks if there are empty clusters, i.e., if $N_k = 0$ for any $k = 1, \dots, K$. In this case, the empty clusters are deleted and K is decremented by the number of empty clusters.

These steps are repeated until convergence or until a maximum number of iterations is reached. There are several ways to test for convergence in VI—in our clustering implementation, we simply check that the cluster means have stopped moving to within some small tolerance [19]. Refer to Algorithm 3.1 for pseudocode describing our algorithm.

3.5 Algorithm demonstration

We demonstrate that a variational BNP clustering algorithm with our PCRCP approximation can reduce over-clustering with a simple 2-d clustering example. Note that this demonstration is not intended to be a thorough study of the algorithm—we provide much deeper analysis for a variety of clustering problems in Chapters 5 to 8. Rather, we now present an illustrative example that also serves to motivate the hyperparameter tuning methods proposed in Chapter 4.

Six cluster centers were selected randomly from an equally spaced 5-by-5 grid ranging from -10 to 10 in each dimension. For each cluster, each dimension was assigned a random variance in the range $[0.8, 1.2]$. Then, N_k points were drawn for each cluster. We used our variational BNP clustering algorithm with the PCRCP to cluster the data for 51 values of r spaced evenly between 1.0 and 5.0 and computed the cluster count estimation error $\Delta K = \hat{K} - K$ for each. To show how the CRP hyperparameter α impacts the clustering solution, we repeat this exercise for $\alpha \in \{0.1, 1.0, 10.0\}$.

This process was repeated for 100 Monte Carlo iterations, each comprising randomly chosen sets of $K = 6$ clusters with the number of points per cluster $N_k \in \{100, 400\}$. The signed error in the estimated number of clusters, ΔK was averaged and plotted in Fig. 3.1. When $\Delta K > 0$ ($\Delta K < 0$), the algorithm estimated too many (few) clusters. When $\Delta K \approx 0$, the number of clusters is correctly estimated. The standard CRP implementation of BNP clustering is obtained for $r = 1.0$, which can be used as a baseline for comparison with our approach when $r > 1.0$.

The results in the figure reveal some interesting properties of our variational BNP clustering algorithm. First, we comment on the PCRCP hyperparameter r , since this is a key component of the novel contribution of this work. The standard variational CRP-based clustering algorithm (i.e., when $r = 1.0$) consistently estimates too many clusters. This tendency persists even when α is made smaller to reduce the probability that a new cluster is generated by the CRP. As r increases, ΔK goes down, showing that the PCRCP is limiting

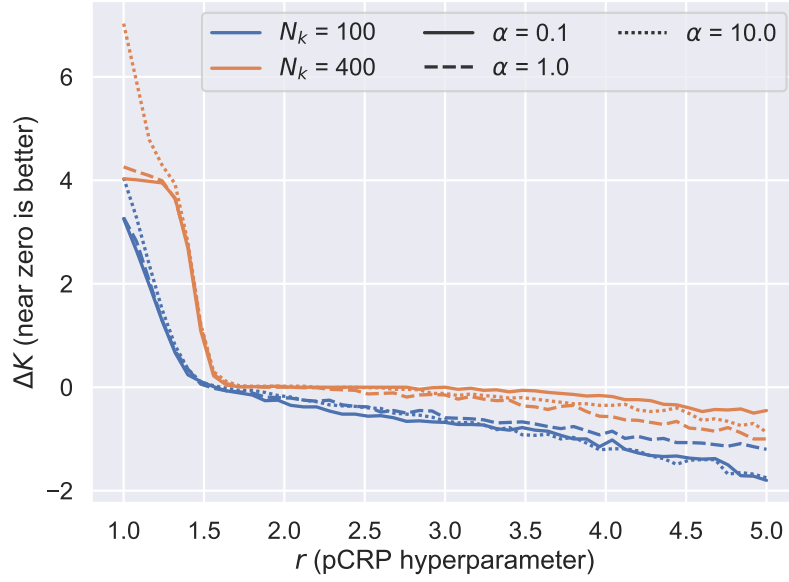


Figure 3.1: Cluster count estimation error ΔK for various values of r , α , and N_k on a six-cluster problem. When r is near one, $\Delta K > 0$, indicating that the number of clusters is over-estimated. As r increases, ΔK decreases, eventually yielding an under-clustered result (i.e., the estimated number of clusters is less than the true number of clusters).

the effect of over-clustering. However, note that the “best” r for this six cluster problem (i.e., when $\Delta K = 0$, indicating that the correct number of clusters has been inferred) for $N_k = 100$ is around $r = 1.5$, whereas for the $N_k = 400$ case, the “best” $r \approx 1.75$. Note also that for $N_k = 400$, the algorithm extracts the correct number of clusters for a larger range of r . Although this indicates that the PCRPP can be effective for reducing overclustering in VI, it is undesirable that the best hyperparameter values may be dependent on the number of data points per cluster because this quantity is not known a priori.

Second, this experiment does not indicate that there is any advantage to tuning both α and r . Specifically, the demonstration here shows that r has a much greater impact on the final estimated number of clusters. Like [29], we choose to fix $\alpha = 1.0$ and control the number of clusters by tuning r alone in the remainder of this work. Clearly, it is critically important to choose a proper value for the PCRPP hyperparameter in order to obtain a clustering solution with the correct number of clusters. To address this, we propose

Algorithm 3.1 Online variational BNP clustering with a Gaussian mixture model.

Require: Datasets $\mathcal{X}^t, t = 1, 2, \dots$

Require: Normal-Wishart hyperparameters $\mu_0, \nu_0, \mathbf{W}_0, \lambda_0$

Require: CRP concentration hyperparameter α , PCRPP hyperparameter r , initial number of clusters K_0

```
1:  $K \leftarrow K_0$ 
2: Initialize  $q^t(\mathcal{Z})$  via  $K$ -means with  $K = K_0$ .
3: for  $t = 1, 2, \dots$  do
4:   repeat
5:     Update  $q^t(\Theta)$  via (2.7)
6:     Update  $q^t(\mathcal{Z})$  via (2.8)
7:     Compute cluster counts  $N_k, k = 1, \dots, K + 1$ 
8:     if  $N_{K+1} > 0$  then
9:       Add a new cluster
10:       $K \leftarrow K + 1$ 
11:    end if
12:    if Any  $N_k = 0$  for  $k = 1, \dots, K$  then
13:       $K \leftarrow K - |\{N_k = 0 : k = 1, \dots, K\}|$ 
14:      Remove clusters  $k$  with  $N_k = 0$ 
15:    end if
16:  until convergence
17: end for
```

some strategies for tuning the hyperparameter in Chapter 4.

3.6 Summary

Standard BNP clustering algorithms tend to over-estimate the number of clusters in the data [27]. The recently-developed PCRPP has helped to resolve this [29], but is limited to offline clustering because it relies on slow MCMC inference. When data arrives in a continuous stream and the number of clusters can change over time, online clustering is necessary. In this chapter, we implemented a new online variational BNP clustering algorithm that reduces over-clustering in the inferred mixture. To do this, we derived a Taylor approximation (3.2) for the PCRPP. We showed how to use this approximation in a BNP GMM and demonstrated the impact of the PCRPP hyperparameter r . In the following chapter, we discuss methods for tuning r without supervision.

Chapter 4

Hyperparameter tuning

As shown in Section 3.5, it is critically important to choose a proper value for the PCRP hyperparameter r in order to obtain a clustering solution with the correct number of clusters. In this chapter, we propose some methods for tuning the hyperparameter. We start by examining some of the empirical properties of the PCRP and discuss strategies for inferring the hyperparameter statistically. For offline hyperparameter tuning, we also propose a root-finding method inspired by Gaussian mixture reduction. We also propose a way to combine these methods in an optimization-based approach. Finally, we extend these techniques to the online setting by employing multivariate change detection and exploiting our efficient variational implementation. The strategies are evaluated for effectiveness on synthetic datasets in Chapter 5, employed for clustering radar pulses in Chapter 6, and demonstrated for neural spike sorting in Chapter 7. The root-finding method is further exercised in Chapter 8.

4.1 Empirical properties of the pCRP hyperparameter

Tuning the concentration parameter α in the standard CRP (2.9) was studied as early as 1995 [87], where a Gibbs sampling implementation was proposed to approximate

$$p(\alpha \mid K, N) \propto p(K \mid \alpha, N)p(\alpha). \quad (4.1)$$

This is straightforward because $p(K \mid \alpha, N)$ is available in closed form [88]:

$$p(K \mid \alpha, N) = |s(N, K)|\alpha^K \frac{\Gamma(\alpha)}{\Gamma(\alpha + N)} \quad (4.2)$$

where K is the number of clusters, N is the number of data points, and $s(N, k)$ is the Stirling number of the first kind. While it is natural to take a similar approach and consider writing $p(K \mid r, \alpha, N)$ for the pCRP, we find that it does not have a similar compact form. This is because the pCRP forfeits exchangeability in favor of greater control over the number of clusters. For example, we could not easily write the denominator of $p(K \mid \alpha, N, r)$ for the pCRP in terms of Gamma functions (or factorials) because the order in which customers arrive at the restaurant matters—each table is weighted *exponentially* by the number of customers at that table.

The number of clusters conditioned on the hyperparameter is still a quantity of interest, however, and its distribution (or an approximation thereof) may be useful in deriving a hyperparameter tuning scheme. We study $p(K \mid r)$ by simulating 150,000 samples from the pCRP for $N \sim \text{Unif}(100, 800)$ and $r \sim \text{Norm}(r_i, 10^{-4})$, with r_i taking 150 evenly-spaced values in the interval $[1.0, 2.5]$. We group values of r into bins of width 10^{-2} , denoted $\mathcal{R}_i$, and compute empirical distributions of $p(K \mid r \in \mathcal{R}_i)$ (examples for $\bar{r}_i \approx 1.0, 1.5, 2.0$ are shown in Fig. 4.1). Note that to simplify our analysis, we draw samples for a large range of N and examine $p(K \mid r)$ without explicitly modeling the influence of N . Additionally, we choose to study the pCRP hyperparameter exclusively and hold $\alpha = 1$ constant, following

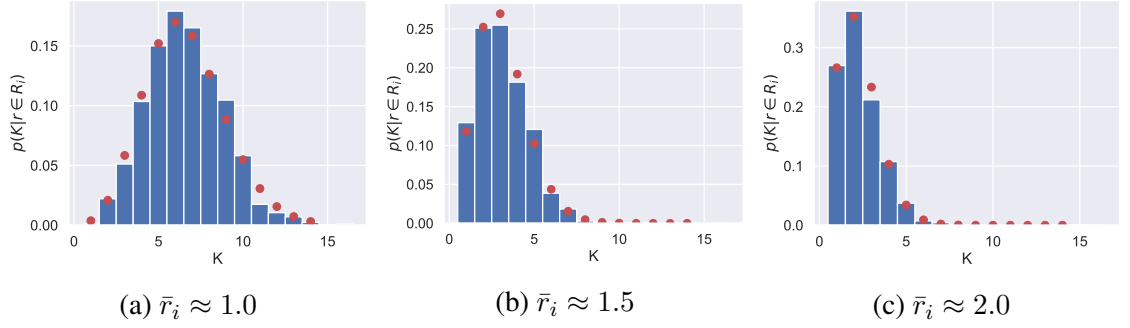


Figure 4.1: Histograms for K given $\mathcal{R}_i$ along with Poisson fits (red dots) for various $\mathcal{R}_i$ bins. The mean of the values in $\mathcal{R}_i$ is denoted $\bar{r}_i$.

the rationale laid out in Chapter 3 and [29].

The empirical distributions shown in Fig. 4.1 confirm the notion that higher values of r tend to favor fewer clusters (smaller K). The shape of the empirical distributions suggests that $K - 1$ can be approximated by a Poisson distribution with rate parameter dependent on r , shown as red dots. An MLE-derived rate parameter $\hat{\xi}_r$ can be found such that $\text{Pois}(K - 1; \hat{\xi}_r)$ approximates the empirical distribution $p(K|r)$. Chi-squared goodness of fit tests resulted in p values less than 0.1 and 0.3 for roughly 75% and 90% of the estimated $\hat{\xi}_r$, respectively, indicating that the Poisson approximation is reasonable for most values of r .

Plotting $\hat{\xi}_r$ as a function of r (Fig. 4.2) reveals that the rate parameter follows an approximately exponential trend given by the equation

$$\xi(r) = 4.13 \exp(-2.79(r - 1.04)) + 1.03, \quad (4.3)$$

derived via non-linear least squares estimation provided by the `scipy` package [89]. The curve fit was verified using an additional holdout set of 50,000 samples of $p(K | r)$ using the PCRPs as described above, with coefficient of determination $R^2 = 0.994$ and root mean-squared error $\text{RMSE} = 0.088$. We found that the level of precision is not improved by maintaining greater than two significant digits in the curve parameters, and the precision

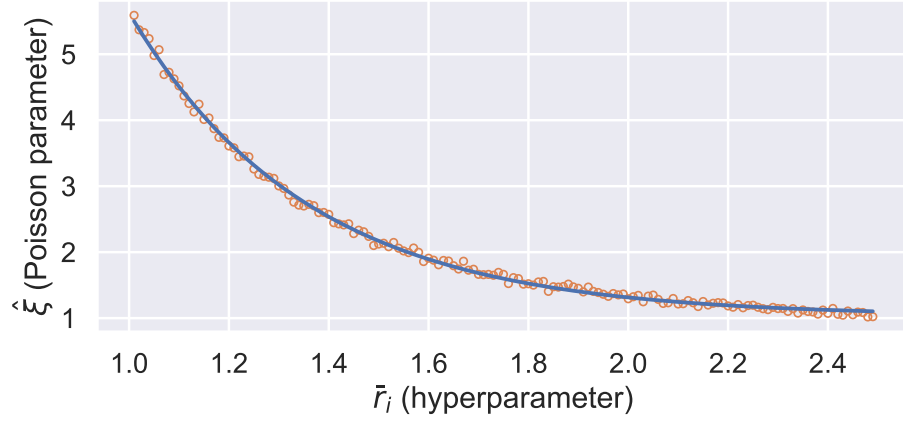


Figure 4.2: Curve fitment for $(\bar{r}_i, \hat{\xi})$ pairs, where $\hat{\xi}$ is the MLE parameter for a Poisson distribution.

is decreased when the parameters are rounded to only one significant digit.

We thus arrive at the following closed-form approximation for the probability of the number of clusters K given the hyperparameter r in the PCRPP:

$$\hat{p}(K \mid r) = \text{Pois}(K - 1; \xi(r)) \quad (4.4)$$

$$= \frac{\xi(r)^{K-1} \exp(-\xi(r))}{(K - 1)!}, \quad (4.5)$$

where $\xi(r)$ is given in (4.3) and we have placed a hat on p to denote that it is an approximation of $p(K \mid r)$ based on empirical analysis. Note that this formulation is specific to the PCRPP and is based purely on empirical analysis. Moreover, it is not necessarily generalizable for other uses of the PCRPP when $r \notin [1.0, 2.5]$ or $N \notin [100, 800]$.

A closed-form equation for a probability density, even if it is approximate, can be useful in inference tasks. In our case, (4.5) even has a conjugate Gamma prior. This enables fast inference by estimating $\hat{\xi}$ from the Gamma posterior hyperparameters and inverting (4.3). However, posterior inference with only a single observation of K per iteration of the algorithm means that we must be very careful when selecting the Gamma prior hyperparameters. They may easily be too influential (i.e., the value of $\hat{\xi}$ becomes nearly deter-

ministic, so we might as well set r by hand) or not influential enough (i.e., the clustering algorithm will converge before enough evidence is gathered to establish a good posterior estimate). The problem of Gamma prior hyperparameter selection is exacerbated by lack of a priori knowledge of the number of clusters in the data. Alternatively, we can choose a non-conjugate prior and infer $p(r \mid K)$ via the Metropolis-Hastings algorithm [90]. For example, an exponential prior over $r - 1$ would reflect that r is bounded below by 1 and that high values of r are increasingly unlikely. This has the drawback of being slow to converge, which makes such an approach unsuitable for online processing.

4.2 Numerical methods for pCRP hyperparameter tuning

In light of the challenges presented by treating hyperparameter tuning as an inference problem, we now present an alternative approach. To reduce over-clustering, we wish to choose r such that the number of clusters is minimized *without compromising the expressiveness of the model*. In other words, we want the model to be compact, yet sufficiently representative of the underlying data. This concept can be described in terms of a simple optimization problem:

$$\begin{aligned} r^* &= \arg \min_r \hat{K}_r \\ \text{s.t. } \varepsilon(\Theta, \hat{\Theta}_r) &\leq \eta, \end{aligned} \tag{4.6}$$

where $\hat{K}_r$ is the number of components resulting from fitting the model with hyperparameter r and ε is some function that measures the error between the true model parameters Θ and the estimated model parameters $\hat{\Theta}_r$. This formulation requires the total error to be no greater than η . A natural choice for an error measure would be $\varepsilon = \text{KL}$, but we do not know the Θ corresponding to the true generative model, and there is no closed form solution for the KL divergence between GMMS, although approximations exist (e.g., [91]). Moreover,

KL divergence is uncalibrated, taking values from 0 to ∞ , making selection of η difficult. Instead of relying on an approximation of the KL divergence between GMMs, we propose an approach inspired by Gaussian mixture reduction.

4.2.1 Gaussian mixture reduction

Gaussian mixture reduction is the process by which the number of clusters in a GMM is reduced while attempting to maintain model expressiveness. Numerous examples can be found in the literature, most of which attempt to reduce a GMM with a known number of clusters K to a pre-specified number of clusters $K' < K$, e.g., [92], [93]. Our problem is similar, yet distinct. We can use our algorithm to estimate K , but we do not know a specific value $K' < K$ to perform canonical Gaussian mixture reduction. Nonetheless, the demonstrated tendency for BNP GMMs to overestimate the number of clusters in the data means that Gaussian mixture reduction may be a good source for inspiration.

In Gaussian mixture reduction, a common measure of the similarity between two GMMs is the integrated squared error (ISE). Equations (4.7) to (4.14) are adapted from [94]–[97]. Let two PDFs over $\mathbf{x} \in X$ be parameterized by Θ and Θ' as $p(\mathbf{x} \mid \Theta)$ and $q(\mathbf{x} \mid \Theta')$ (not necessarily the generative model p and variational approximation q from previous discussion). The ISE has the following generic form:

$$\text{ISE}(\Theta, \Theta') = \int_{\mathcal{X}} (p(\mathbf{x} \mid \Theta) - q(\mathbf{x} \mid \Theta'))^2 d\mathbf{x} \quad (4.7)$$

where, for collapsed GMMs,

$$p(\mathbf{x} \mid \Theta) = \frac{1}{K} \sum_{k=1}^K \text{Norm}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \quad (4.8)$$

$$q(\mathbf{x} \mid \Theta') = \frac{1}{K'} \sum_{k=1}^{K'} \text{Norm}(\mathbf{x}; \boldsymbol{\mu}'_k, \boldsymbol{\Sigma}'_k). \quad (4.9)$$

In these equations we have used the prime symbol $(\cdot)'$ to denote parameters taken from Θ' ; parameters without the prime symbol are taken from Θ . $\text{Norm}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the Gaussian PDF with covariance matrix $\boldsymbol{\Sigma}$ evaluated at $\mathbf{x}$. In this case, the ISE between two GMMs with different parameters can be computed in closed-form as

$$\text{ISE}(\Theta, \Theta') = J_{pp} - 2J_{pq} + J_{qq} \quad (4.10)$$

where

$$J_{pp} = \frac{1}{K^2} \sum_{k=1}^K \sum_{\ell=1}^K \text{Norm}(\boldsymbol{\mu}_k; \boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_k + \boldsymbol{\Sigma}_\ell) \quad (4.11)$$

$$J_{pq} = \frac{1}{KK'} \sum_{k=1}^K \sum_{\ell=1}^{K'} \text{Norm}(\boldsymbol{\mu}_k; \boldsymbol{\mu}'_\ell, \boldsymbol{\Sigma}_k + \boldsymbol{\Sigma}'_\ell) \quad (4.12)$$

$$J_{qq} = \frac{1}{(K')^2} \sum_{k=1}^{K'} \sum_{\ell=1}^{K'} \text{Norm}(\boldsymbol{\mu}'_k; \boldsymbol{\mu}'_\ell, \boldsymbol{\Sigma}'_k + \boldsymbol{\Sigma}'_\ell). \quad (4.13)$$

The ISE can be normalized such that it is bounded on the interval $[0, 1]$. The normalized ISE is given by [94], [95]:

$$\text{NISE}(\Theta, \Theta') = \frac{\text{ISE}(\Theta, \Theta')}{J_{pp} + J_{qq}}. \quad (4.14)$$

Note that NISE can be efficiently computed directly from the GMM parameters and does not require any data.

4.2.2 Analyzing NISE for redundant models

Recall that in general, as the PCR hyperparameter r increases in our variational BNP clustering algorithm, the number of clusters in the resulting model tends to decrease. Increasing r can have the general effect of reducing over-clustering or redundancy in the model. For our work, we want to understand how the NISE reflects similarity between a GMM and an over-clustered version of that GMM so we can use it to select a good hyperparameter value.

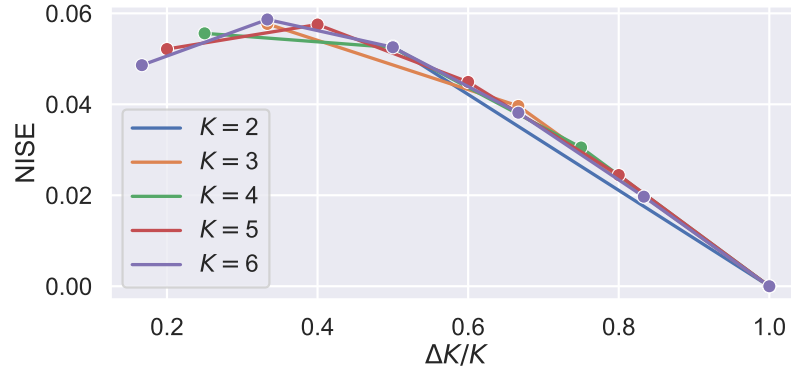


Figure 4.3: $\text{NISE}(\Theta, \Theta')$ vs $\Delta K/K$, where K is the number of clusters in the true mixture Θ and Θ' is a redundant version of Θ with $K' > K$ clusters. The NISE peaks near 0.06 for all values of K .

We now study how the NISE responds to over-clustering by considering two GMMs parameterized by Θ and Θ' with K and $K' > K$ clusters, respectively. To simplify the analysis, assume that the GMM parameterized by Θ is the “true” mixture and that the over-clustered GMM parameterized by Θ' contains duplicates of the true clusters with a slight perturbation in the mean. In other words, Θ' is essentially a redundant version of Θ . For $K = 2, \dots, 6$, we randomly remove clusters from the redundant mixture one-by-one (starting with $K' = 2K$) and compute $\text{NISE}(\Theta, \Theta')$ according to (4.14). The NISE values are averaged for each value of K and K' across 100 Monte Carlo iterations. The result is plotted in Fig. 4.3 with respect to $\Delta K/K = K'/K - 1$. Note that $\Delta K/K = 1$ corresponds to a completely redundant model with $K' = 2K$ and as $\Delta K/K$ decreases, the number of redundant clusters is reduced.

We first note that $\text{NISE}(\Theta, \Theta') = 0$ in the most redundant case when $\Delta K/K = 1$. Studying the definition of the NISE reveals that a mixture model that contains K clusters and an integer number of duplicates for each cluster will result in $\text{NISE} = 0$. However, this situation rarely arises in practice. Moreover, the tuning method we propose in the following section does not seek to minimize NISE, so this does not necessarily present a problem. As duplicate clusters are successively removed (decreasing $\Delta K/K$), the NISE climbs steadily

and consistently for various numbers of “true” clusters K . Clearly for each value of K , the maximum NISE is approximately 0.06 and occurs when the redundant model Θ' is nearly reduced to $\Theta' = \Theta$.

4.2.3 Employing the NISE to tune r

Because the PCRPP is designed to prevent over-clustering, we wish to choose the hyperparameter r to minimize the number of components in the resulting model. However, we should choose r such that the final model is sufficiently expressive—if the inferred number of clusters is too small, the model will not provide a good description of the data. Empirically we have seen that when $r = 1.0$, corresponding to the standard CRP, there is a tendency to overestimate the number of clusters (see the example at the end of Chapter 3), but the resulting model still covers the support of the data. In other words, the overclustered model is *expressive*, but *redundant*. Solving the optimization problem in (4.6) would resolve this, but since we do not know the true parameters Θ of the underlying data, we cannot compute $\varepsilon(\Theta, \hat{\Theta}_r)$. However, we can minimize the number of clusters found in the data subject to a constraint on the *difference in expressiveness* of the model relative to the over-clustered case, measured by the NISE. Such an approach does not correct errors with respect to the true model parameters, but it can help ensure that a hyperparameter is selected that yields a compact model that maintains the expressiveness of the over-clustered model.

We thus tune r via a surrogate optimization problem:

$$\begin{aligned} r^* &= \arg \min_r \hat{K}_r \\ \text{s.t. } &\text{NISE}(\hat{\Theta}_{r_0}, \hat{\Theta}_r) \leq \eta, \end{aligned} \tag{4.15}$$

where $\hat{\Theta}_{r_0}$ represents a BNP GMM fit with $r = r_0$ prior to (and held constant during)

optimization and $\hat{\Theta}_r$ is the reduced GMM with $r < r_0$. For the work presented here, we fix $r_0 = 1.0$. Because we use NISE as an error measure, we know that $\eta \in [0, 1]$ and can choose this value empirically. Since $\hat{K}_r$ generally decreases monotonically as a function of r (see the example in Section 3.5), the optimization problem can be solved efficiently by finding the value of r that satisfies the constraint with equality, i.e. r^* is the root of

$$\text{NISE}(\hat{\Theta}_{r_0}, \hat{\Theta}_{r^*}) - \eta = 0. \quad (4.16)$$

If such a root cannot be found, we make the assignment $r^* = r_{\max}$. With the exception of the analysis in the following section, we set $r_{\max} = 2$ to improve the convergence of the root-finding routine. Note also that once $\hat{\Theta}_{r_0}$ is obtained, we can get $\hat{\Theta}_r$ via an efficient online update, so the cost involved with solving (4.16) is reduced. In this work, we refer to this hyperparameter tuning method as the *root-finding* method.

4.2.4 Selecting η

In Section 3.5 it was shown that the number of clusters obtained in the final mixture is sensitive to the value of r , the PCRP hyperparameter. We now examine the effects of η , the root-finding hyper-hyperparameter which controls the value of r based on the solution to (4.16).

Fig. 4.4 depicts the ΔK for datasets with the number of points per cluster $N_k \in \{100, 400\}$, generated as described in Section 3.5. We also show the value of r that was chosen by the root-finding method. Note that for these plots we have used $r_{\max} = 5$ to examine algorithm performance beyond the most generally effective values $1 \leq r \leq 2$. Results are given as averages over 100 Monte Carlo iterations. Curves in the left and right columns are results obtained for 30 equally-spaced values of η in the range $[0.01, 0.3]$ and $[0.01, 0.1]$, respectively. The boxes in the left column correspond to the plotted regions in the right column.

We found that the best ΔK (near zero) occurs around $\eta = 0.05$ for $N_k = 100$ and $\eta = 0.07$ for $N_k = 400$. This dependency between η and N_k is similar to the dependency between r and N_k that was seen in Section 3.5. However, the root-finding method for automatically tuning r has several advantages over selecting r manually. First, η is interpretable because it is related directly to the NISE. The meaning of the PCR-P hyperparameter r , on the other hand, is difficult to discern. Second, η is bounded below and above, while r is only bounded from below, making it somewhat easier to select with little a priori knowledge. Finally, using an algorithm to tune the PCR-P hyperparameter r means that the algorithm may be more capable to adapt to changes in dynamic datasets—the analysis made here and in Section 3.5 provide limited insight into the performance of the algorithm on infinitely variable online clustering tasks. We demonstrate the effectiveness of root-finding tuning in Chapters 5 to 8.

As a result of this analysis, we find that the best approach is to set $\eta < 0.1$. We choose to conservatively fix $\eta = 0.06$ in this work and do not adjust it for different experiments in order to avoid overfitting η . This value of η occurs between the $\Delta K = 0$ location of the $N_k = 100$ and $N_k = 400$ lines in Fig. 4.4.

4.3 Combining inference-based and numerical approaches

As mentioned in Section 4.1, direct use of (4.5) for statistical inference is not feasible. Although we will later show that the root-finding approach is effective for tuning the hyperparameter r , it relies on the selection of a *hyper*-hyperparameter η which controls the allowable “error” incurred by reducing the GMM. Selecting this hyperparameter is arguably easier than choosing r directly because it is less dependent on the data, but it is desirable to have a tuning method that does not rely on any hyper-hyperparameters at all.

To this end, we propose a second approach we call the *optimization*-based method.

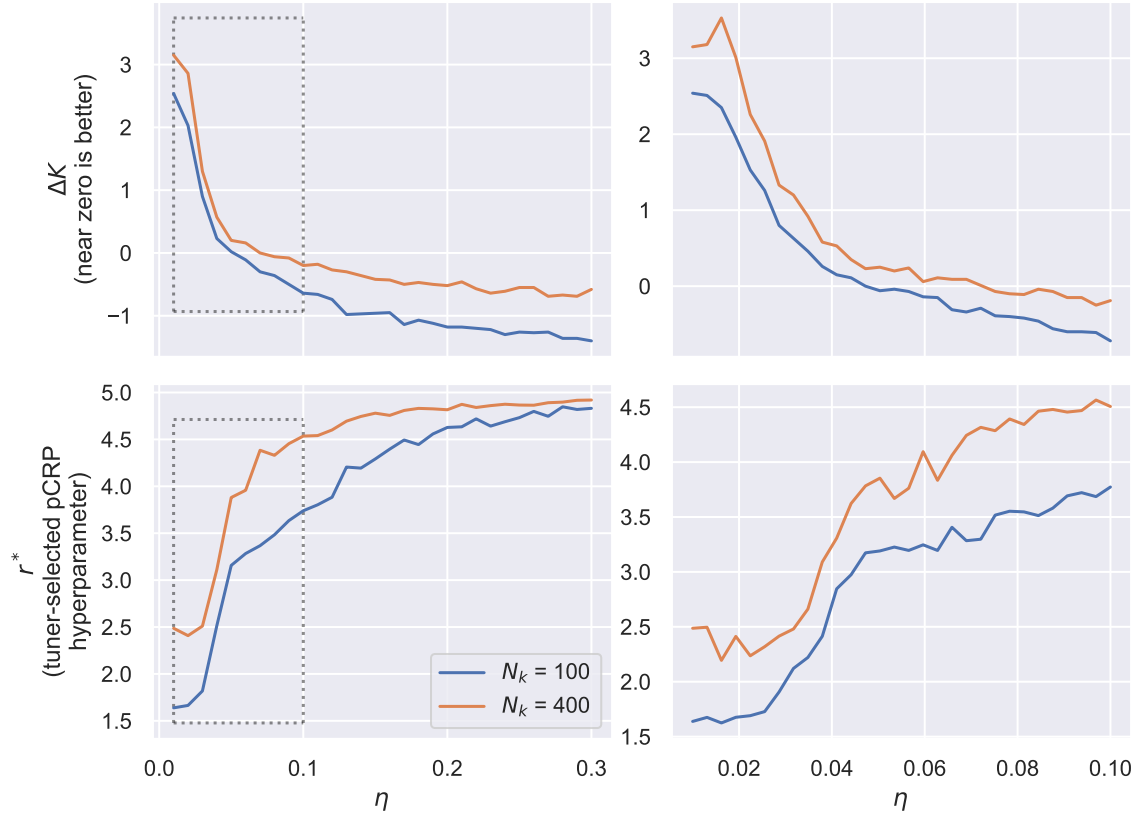


Figure 4.4: Cluster count estimation error ΔK and the value of r (PCR hyperparameter) returned by the root-finding method for the number of points per cluster $N_k \in \{100, 400\}$. Boxes in the figures on the left correspond to the plotted region on the right. Note that for clarity of presentation, the y-axes in the left and right columns are not aligned. These plots indicate that ΔK is nearest zero for approximately $0.05 \leq \eta \leq 0.07$. Note that the value of r returned by the root-finding method for some η is not the same for different values of N_k . Left column: $\eta \in [0.01, 0.3]$. Right column: $\eta \in [0.01, 0.1]$.

This method optimizes

$$r^* = \arg \max_r \left[(1 - \text{NISE}(\hat{\Theta}_{r_0}, \hat{\Theta}_r)) \hat{p}(K \mid r) \right] \quad (4.17)$$

subject to a constraint that $1 \leq r \leq r_{\max}$, where $\hat{p}(K \mid r)$ is given by (4.5).¹ The objective function leverages the fact that the NISE is bounded on the interval $[0, 1]$ like a probability density. Thus, its contribution to the overall objective function value is balanced with respect to $\hat{p}(K \mid r)$.

Intuitively, the objective function is near 1 when $\text{NISE}(\hat{\Theta}_{r_0}, \hat{\Theta}_r) \rightarrow 0$, indicating that $\hat{\Theta}_r$ is “similar” to $\hat{\Theta}_{r_0}$ in the NISE-sense, and $\hat{p}(K \mid r)$ is maximum, indicating that the number of clusters is well-matched to the value of r . The objective function is near 0 when either $\text{NISE}(\hat{\Theta}_{r_0}, \hat{\Theta}_r) \rightarrow 1$, indicating a mismatch between the models, or $\hat{p}(K \mid r)$ is low, indicating that the current value of r is not likely to produce the number of clusters being estimated.

Compared to the root-finding approach in Section 4.2.3, the optimization approach does not rely on any hyper-hyperparameter selection on the user’s part. The model for $\hat{p}(K \mid r)$ was found empirically in Section 4.1, and $r_{\max}$ can be safely fixed at 2 without negatively impacting the result. This is a desirable property when it is not known a priori how similar the reduced GMM should be compared to the over-clustered one (quantified by η in (4.16)). However, this property comes at the cost of increased computational complexity—we demonstrate this in Chapter 5.

4.4 Tuning the hyperparameter online

As a dynamic dataset is processed, the number of clusters in the data can change over time. We have observed that when clusters in the data disappear, the algorithm generally responds

¹Although $r_{\max}$ is *technically* a hyper-hyperparameter, it does not need to be tuned—it merely provides an upper bound on r to speed up convergence in the optimization routine.

appropriately by removing the corresponding component. However, when the data exhibits new clusters, the algorithm tends to increase the covariance of an existing GMM component rather than add a new one. This means that in certain cases, the algorithm may begin to under-estimate the number of clusters (i.e., $\Delta K < 0$). One way to resolve this issue is to re-fit the model with a new hyperparameter using one of the previously described tuning strategies [98].

A naive solution is to simply re-fit the model from scratch for every new minibatch. While this may resolve the ΔK problem, the tuning procedure can be costly and should only be invoked when necessary. Instead, we employ efficient multivariate change detection to determine if a new minibatch is likely to contain new clusters. If new clusters are present, a re-fit is triggered. Otherwise, the model is updated online as usual without re-fitting [98].

For change detection, we implement the semi-parametric log likelihood (SPLL) approach described in [99]. Before clustering minibatch $\mathcal{X}^t$, we compute the test statistic

$$\frac{1}{|\mathcal{X}^t|} \sum_{\mathbf{x} \in \mathcal{X}^t} (\mathbf{x} - \boldsymbol{\mu}_*^{t-1})^\top \mathbf{T}_*^{t-1} (\mathbf{x} - \boldsymbol{\mu}_*^{t-1}) \underset{\text{no change}}{\overset{\text{change}}{\geq}} D + \sqrt{2D}, \quad (4.18)$$

where $\boldsymbol{\mu}_*^{t-1}$ and $\mathbf{T}_*^{t-1}$ are the mean and precision matrix for the cluster nearest to $\mathbf{x}$ in the previous minibatch and D is the data dimensionality. The decision “change” means that the model should be re-fit for $\mathcal{X}^t$ with a new hyperparameter, while the decision “no change” means that the model can be updated in an online fashion without changing the hyperparameter. The decision criterion $(D + \sqrt{2D})$ arises from the fact that the sum of the squared Mahalanobis distances is chi-squared distributed with D degrees of freedom with mean D and standard deviation $\sqrt{2D}$. Hence the SPLL approach declares a change when the test statistic is greater than one standard deviation above the mean [99].

We acknowledge that the literature contains many examples of change detection that could be useful for this purpose. However, a full investigation of change detection methods

is out of scope for this work, and we have found empirically that the SPLP approach is effective and efficient.

Refer to lines 3–10 in Algorithm 4.1 to see how hyperparameter tuning and change detection are incorporated into our algorithm.

Algorithm 4.1 Online variational BNP clustering with a Gaussian mixture model and automatic hyperparameter tuning.

Require: Datasets $\mathcal{X}^t, t = 1, 2, \dots$

Require: Normal-Wishart hyperparameters $\mu_0, \nu_0, \mathbf{W}_0, \lambda_0$

Require: CRP concentration hyperparameter α , pCRP hyperparameter r , initial number of clusters K_0

```

1:  $K \leftarrow K_0$ 
2: for  $t = 1, 2, \dots$  do
3:   if  $t > 1$  then
4:     Perform SPLP test (4.18)
5:   end if
6:   if  $t = 1$  or SPLP is triggered for minibatch  $t$  then
7:     Initialize  $q^t(\mathcal{Z})$  via  $K$ -means with  $K = K_0$ .
8:     Select hyperparameter  $r$  via root-finding or optimization-based routine
9:   else
10:    Initialize  $q^t(\mathcal{Z})$  using  $q^{t-1}(\Theta)$  via (2.8)
11:   end if
12:   repeat
13:     Update  $q^t(\Theta)$  via (2.7)
14:     Update  $q^t(\mathcal{Z})$  via (2.8)
15:     Compute cluster counts  $N_k, k = 1, \dots, K + 1$ 
16:     if  $N_{K+1} > 0$  then
17:       Add a new cluster
18:        $K \leftarrow K + 1$ 
19:     end if
20:     if Any  $N_k = 0$  for  $k = 1, \dots, K$  then
21:        $K \leftarrow K - |\{N_k = 0 : k = 1, \dots, K\}|$ 
22:       Remove clusters  $k$  with  $N_k = 0$ 
23:     end if
24:   until convergence
25: end for

```

4.5 Summary

Our algorithm implements an approximation of the PCRCP to reduce redundancy in online BNP clustering. The PCRCP makes use of a hyperparameter r that controls the tendency to create more or fewer clusters in the model. The experiment in Section 3.5 showed that proper selection of r is critical for good performance. In this chapter, we proposed two methods for tuning this critical hyperparameter.

We started by presenting an empirical analysis of the PCRCP. We found that the distribution of the number of clusters K , conditioned on the hyperparameter r , can be approximated by a Poisson distribution where the Poisson parameter ξ has an exponential relationship to r (see (4.5)). Although this model would ideally be used for directly inferring the best hyperparameter value given K , we found that hyper-hyperparameter selection and/or restrictions on the inference method prevent this in practice.

Alternatively, we proposed tuning methods that leverage the NISE measure between two collapsed GMMs given in (4.14). The first method, called the root-finding method, finds a hyperparameter value r^* that achieves $\text{NISE}(\hat{\Theta}_{r_0}, \hat{\Theta}_{r^*}) = \eta$, where η is a hyper-hyperparameter between 0 and 1, and $r_0 = 1.0$ to help ensure that $\hat{\Theta}_{r_0}$ is over-clustered. This method does not require hyper-hyperparameter selection, but relies on an empirical analysis of the PCRCP. The second method combines the NISE (4.14) with our empirical model (4.5) by optimizing the objective function (4.17). This method does not require a user-specified hyper-hyperparameter, but it does rely on our empirical analysis. We evaluate these tuning methods in Chapters 5 to 7 and employ the root-finding method in Chapter 8.

Chapter 5

Experiments with synthetic data

To illustrate the properties of our proposed algorithm and show how it improves upon the state of the art in online BNP clustering, we designed two online clustering problems: *rotating* and *radial* online clustering, described in Sections 5.2 and 5.3. These problems are designed to be challenging for online model-based clustering algorithms like the one developed here. In particular, the datasets stress the algorithm’s ability to cope with a randomly changing number of clusters that move dynamically through feature space. The problems are also designed for straightforward performance evaluation—cluster means are not permitted to overlap in feature space, helping to ensure that the underlying model is recoverable, thus providing a fair comparison of algorithm outputs to ground truth cluster labels.

In this chapter, we present results comparing our algorithm with the standard online variational BNP GMM. We show that our approach can obtain good quality clustering solutions with the correct number of clusters, while the standard approach does not. We also show that change detection helps to ensure that our algorithm can adapt to dynamically changing data.

Table 5.1: Prior hyperparameters for the Normal-Wishart data model in synthetic data clustering tasks. A $D \times D$ identity matrix is denoted in this table by $\mathbf{I}$, where D is the data dimensionality. A vector of D zeros is denoted by $\mathbf{0}$.

Hyperparameter	Value
$\mathbf{m}_0$	$\mathbf{0}$
ν_0	0.1
$\mathbf{W}_0$	$0.1\mathbf{I}$
λ_0	D

5.1 Methodology and metrics

The four algorithm configurations listed in Table 5.2 were evaluated for each problem. Two configurations utilize our novel implementation of online variational inference with the PCR-P BNP prior described in Algorithm 4.1. The method without any tuning or change detection (called “None” in these results) is our baseline—it is an online collapsed variational BNP GMM with a CRP prior. We illustrate the effects of change detection alone with the method called “None (w/ CD).” In this case, the algorithm is re-fitted from a K -means initialization with $K = K_0 = 10$ whenever the SPL change detection is triggered. The root-finding and optimization based methods (“Root” and “Opt”) represent our main contributions—online variational BNP clustering that is resistant to over-clustering. For root-finding, we set $\eta = 0.06$. For all methods, the initial number of clusters is $K_0 = 10$. Table 5.1 lists the prior hyperparameters for the Normal-Wishart data model. The precision matrix is constrained to be diagonal.

The algorithms are evaluated for cluster label agreement with ground truth via adjusted mutual information (AMI). AMI is computed for two cluster partitions $\mathcal{U}$ and $\mathcal{V}$ from the equation

$$\text{AMI}(\mathcal{U}, \mathcal{V}) = \frac{\text{MI}(\mathcal{U}, \mathcal{V}) - \mathbb{E}[\text{MI}(\mathcal{U}, \mathcal{V})]}{\max(H(\mathcal{U}), H(\mathcal{V})) - \mathbb{E}[\text{MI}(\mathcal{U}, \mathcal{V})]}, \quad (5.1)$$

where $\text{MI}(\mathcal{U}, \mathcal{V})$ is the estimated mutual information between the partitions and $\mathbb{E}[\text{MI}(\mathcal{U}, \mathcal{V})]$

Table 5.2: Algorithm configurations for experiments with synthetic data. “Shorthand” corresponds to the names used in plots. Our primary contributions are denoted in **boldface text**.

Tuning method	Shorthand	Cluster prior	Comment
None ($r = 1.0$)	None	CRP	Baseline approach (no change detection, fixed $r = 1.0$).
None ($r = 1.0$)	None (w/ CD)	CRP	SPLL change detection only (fixed $r = 1.0$).
Root-finding	Root	PCRCP	Re-tuned when SPLL detects a change.
Optimization	Opt	PCRCP	Re-tuned when SPLL detects a change.

is its expected value (for more details, see [100]). When $\text{AMI} \rightarrow 1$, the clustering solution $\mathcal{U}$ is very similar to $\mathcal{V}$. Conversely, when $\text{AMI} \rightarrow 0$, $\mathcal{U}$ and $\mathcal{V}$ are mutually independent. Thus a correct clustering solution achieves an AMI near 1.

The approximate KL divergence from the true model is also reported. In particular, we compute the symmetrized KL divergence between the true generative model p and the estimated model q defined by $\text{KL}_{\text{sym}}(p \parallel q) = \text{KL}(p \parallel q) + \text{KL}(q \parallel p)$. Because the KL divergence between two GMMs is not known in closed form, we approximate KL_{sym} by Monte-Carlo simulation as suggested by [91] and denote the approximation KL_{MC} . When $\text{KL}_{\text{MC}} \rightarrow 0$, the true and estimated models are similar, with increasing values of KL_{MC} indicating greater dissimilarity.

The error in the number of clusters is measured by the signed deviation from the true number of clusters $\Delta K = \hat{K} - K$, where $\hat{K}$ and K are the estimated and true number of clusters, respectively. When $\Delta K > 0$ ($\Delta K < 0$), the algorithm estimated too many (few) clusters.

Finally, the raw execution time is also provided so that the different tuning schemes can be compared in terms of computational complexity. Datasets were divided into 1000 sequential minibatches. Metrics of performance are computed at each minibatch and averaged. We also report the performance metrics aggregated over an entire dataset by averaging their values across all minibatches.

5.2 Rotating dataset

A *rotating* dataset is a two dimensional dataset comprising K Gaussian clusters with means spaced evenly on a circle of radius R around the origin with equal diagonal covariance matrices. At each timestep t , the cluster means advance around the circle by $\Delta\theta$ degrees. To test the algorithm’s ability to adapt to changes in the number of clusters, at each timestep t , cluster k is removed independently with probability P_{remove} . In subsequent timesteps, the cluster can reappear with probability $P_{\text{replace}|\text{remove}}$. Multiple clusters can be removed or replaced in a single timestep. As new minibatches are sampled from the dataset, the cluster means move around a circular “track” as shown in Fig. 5.1.

This type of dataset stresses an algorithm in the following ways:

- The algorithm must be able to compute/update cluster parameters that are changing as a function of time.
- The algorithm must be able to add and remove clusters according to the data.

Algorithms are evaluated by averaging 100 Monte-Carlo experiments for $K = 8$ clusters with $P_{\text{remove}} = P_{\text{replace}|\text{remove}} = 0.5$ and $\Delta\theta = 0.1$ radians per minibatch. Each cluster contains 100 points. Bar plots indicating performance averaged over all minibatches are given in Fig. 5.2. An interesting result is that for the baseline approach (untuned CRP) the average $\Delta K < 0$. This indicates that the algorithm tends to under-estimate the number of clusters, despite the general tendency for the CRP to cause over-clustering. Fig. 5.3 depicts the Monte-Carlo average ΔK and the true number of clusters K on a per-minibatch basis. These plots reveal that the baseline algorithm generally over-estimates the number of clusters at first (i.e., $\Delta K > 0$), but when the true number of clusters begins to change, estimated clusters are merged erroneously, resulting in a drop in ΔK . When SPL change detection is included without tuning (“None (w/ CD)”), the number of clusters remains steady, but is consistently over-estimated. This behavior is expected from standard CRP implementa-

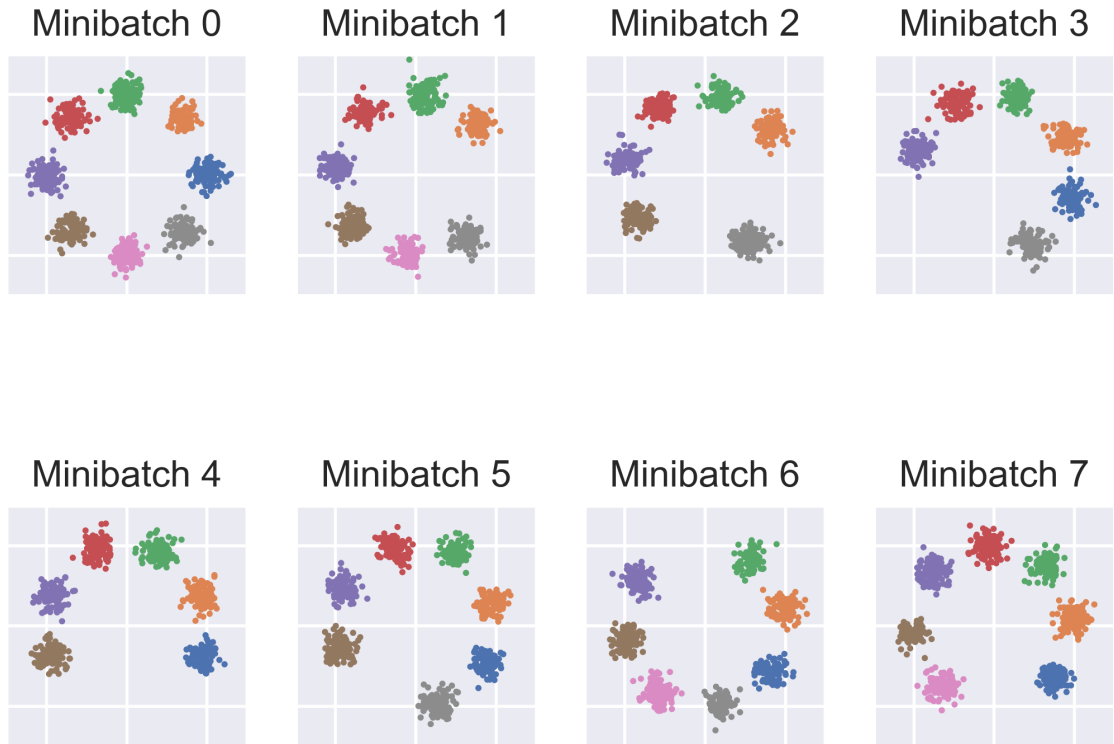


Figure 5.1: Example data from eight consecutive minibatches of a rotating dataset with $K = 8$, $P_{\text{remove}} = 0.75$, $P_{\text{replace}|\text{remove}} = 0.5$, and $\Delta\theta = 0.1$ radians per minibatch. Clusters are colored to help the reader associate consecutive minibatches.

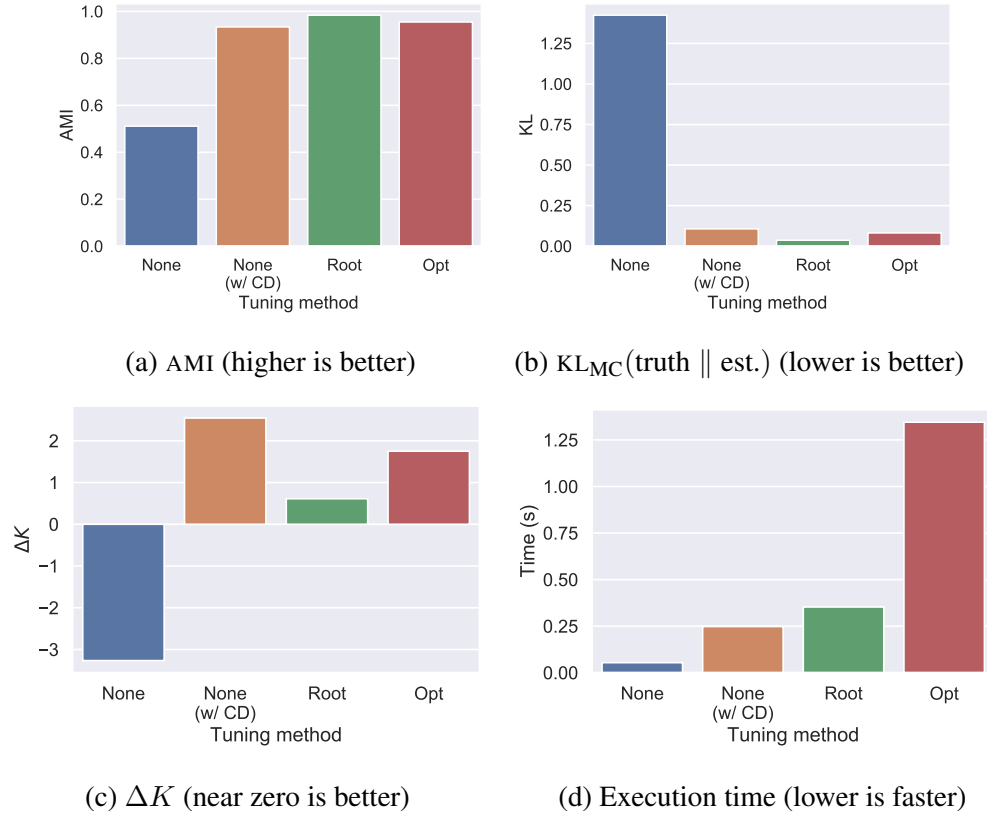


Figure 5.2: Performance metrics for each of the algorithm configurations given in Table 5.2 when tested on the rotating dataset. Metrics were computed for each minibatch, averaged across all Monte-Carlo experiments, and then averaged across minibatches.

tions and serves as motivation for our work. Our algorithm that leverages the tuned PCR with change detection, the primary contribution of this research, performs best by detecting changes in the data stream and re-fitting the algorithm with a better value of r . In general, the root-finding approach for tuning r is faster and provides better cluster assignments and cluster count estimation, measured by the AMI and ΔK , respectively.

5.3 Radial dataset

A *radial* dataset is a two dimensional dataset comprising K Gaussian clusters with means and covariances chosen randomly as described in the example given in Section 3.2. At

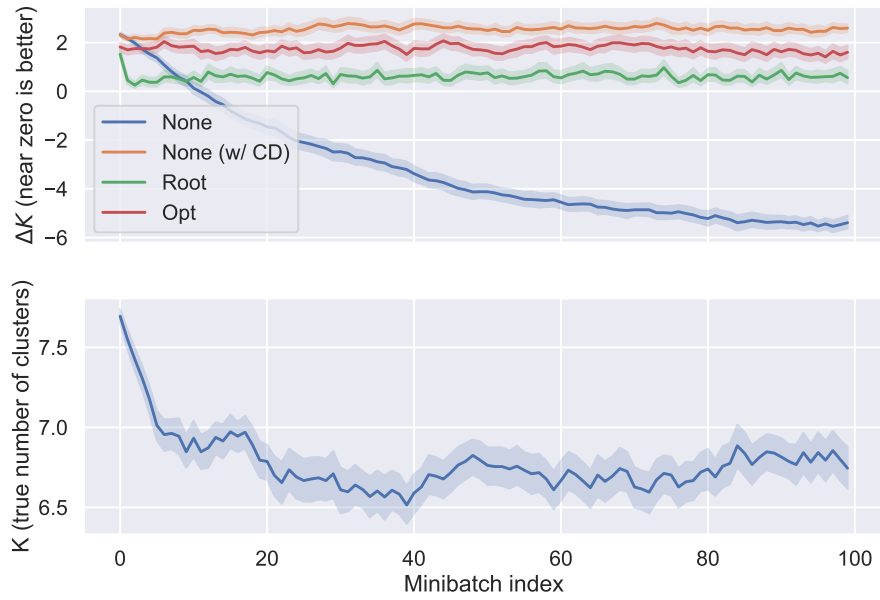


Figure 5.3: Top figure: average per-minibatch ΔK for each of the algorithm configurations given in Table 5.2 when tested on the rotating dataset. Bottom figure: average per-minibatch true number of clusters K in the rotating dataset. Both figures: error regions depict the 95% confidence interval. Note that the true number of clusters is random, thus the true number of clusters has nonzero variance.

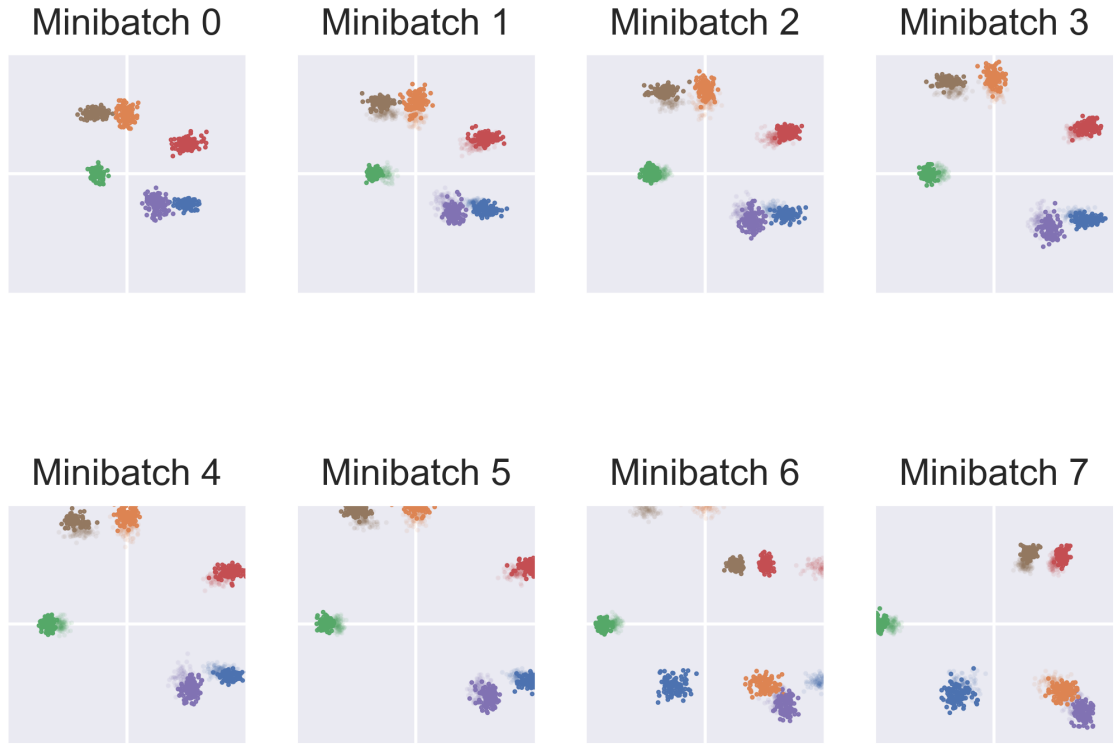


Figure 5.4: Example data from eight consecutive minibatches of a radial dataset with $K = 6$, $R = 1.5\sqrt{200}$, and $S = 2$. Clusters are colored to help the reader associate consecutive minibatches. In the figure for the i -th minibatch, the clusters from the previous minibatch are plotted with some transparency. Note that after minibatch 5, the brown, orange, red, and blue clusters have reached the threshold distance from the origin and new clusters have appeared in minibatch 6.

each timestep t , the cluster means move away from the origin radially with speed S as shown in Fig. 5.4. When a cluster reaches a distance R from the origin, it disappears and a new cluster is created by selecting a new mean from the set and drawing a new random covariance matrix.

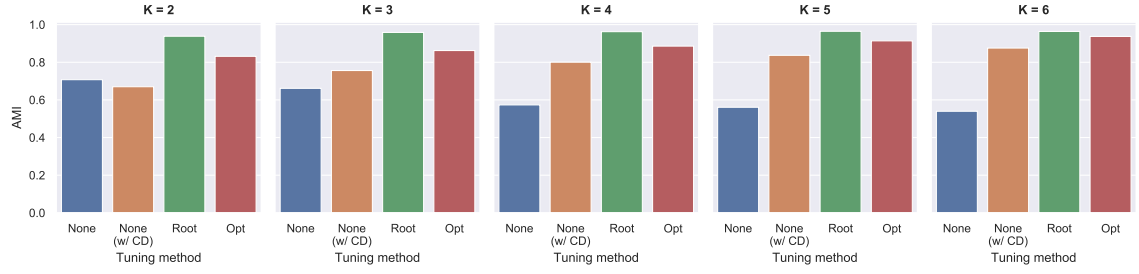
This type of dataset stresses an algorithm in the following ways:

- The algorithm must be able to compute/update cluster parameters that are changing as a function of time.
- The number of clusters in the data is constant, but an algorithm must be able to re-allocate or otherwise account for clusters that “jump” from one location to another

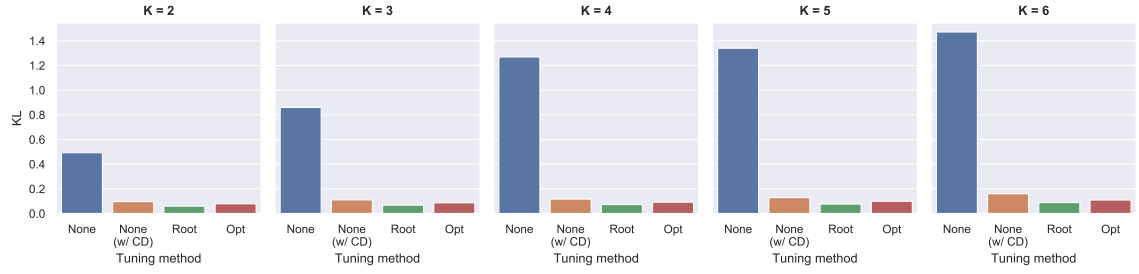
without erroneously merging clusters.

Algorithms are evaluated by averaging 100 Monte-Carlo experiments for each $K = 2, \dots, 6$ clusters with $S = 0.5$ and $R = 1.5\sqrt{200}$. Each cluster contains 100 points. We present results for $K = 2, \dots, 6$. Bar plots indicating performance averaged over all minibatches for each value of K are given in Fig. 5.5. The proposed algorithm, tuned by either the root-finding or optimization approach, outperforms the baseline online CRP (tuning method “None”) in all metrics except for execution time. Like with the rotating data experiments, the relatively smaller AMI and larger KL for the baseline approach indicates that the reported average ΔK must be artificially near zero. This is verified in Fig. 5.6, which depicts the per-minibatch ΔK . The baseline approach over-estimates the number of clusters at first, then under-estimates it by merging clusters that should not be merged. For this experiment, our tuned PCR-based algorithms outperform the baseline. The root-finding approach performs better than the optimization approach in finding the best r to make $\Delta K \approx 0$.

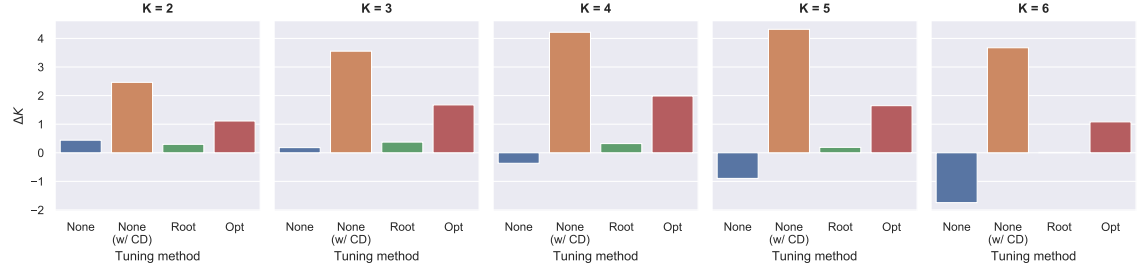
The occasional peaks in ΔK for the root-finding approach shown in Fig. 5.6 indicate that when a re-fit is triggered by SPL, the number of clusters is initially over-estimated before decreasing after a few minibatches. The optimization approach generally estimates more clusters, but is more consistent across minibatches. Without tuning, the baseline approach initially overestimates the number of clusters before erroneously merging existing clusters with new data. Change detection alone fixes this behavior, but does not address over-clustering (see Fig. 5.5c). Our novel combination of the PCR with online processing and change detection addresses both of these issues.



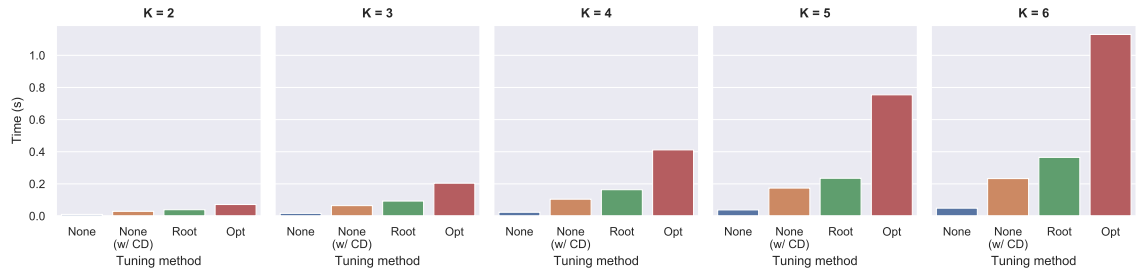
(a) AMI (higher is better)



(b) $KL_{MC}(\text{truth} \parallel \text{est.})$ (lower is better)



(c) ΔK (near zero is better)



(d) Execution time (lower is faster)

Figure 5.5: Performance metrics for each of the algorithm configurations given in Table 5.2 when tested on the radial dataset when $K = 2, \dots, 6$. Metrics were computed for each minibatch, averaged across all Monte-Carlo iterations, and then averaged across minibatches.

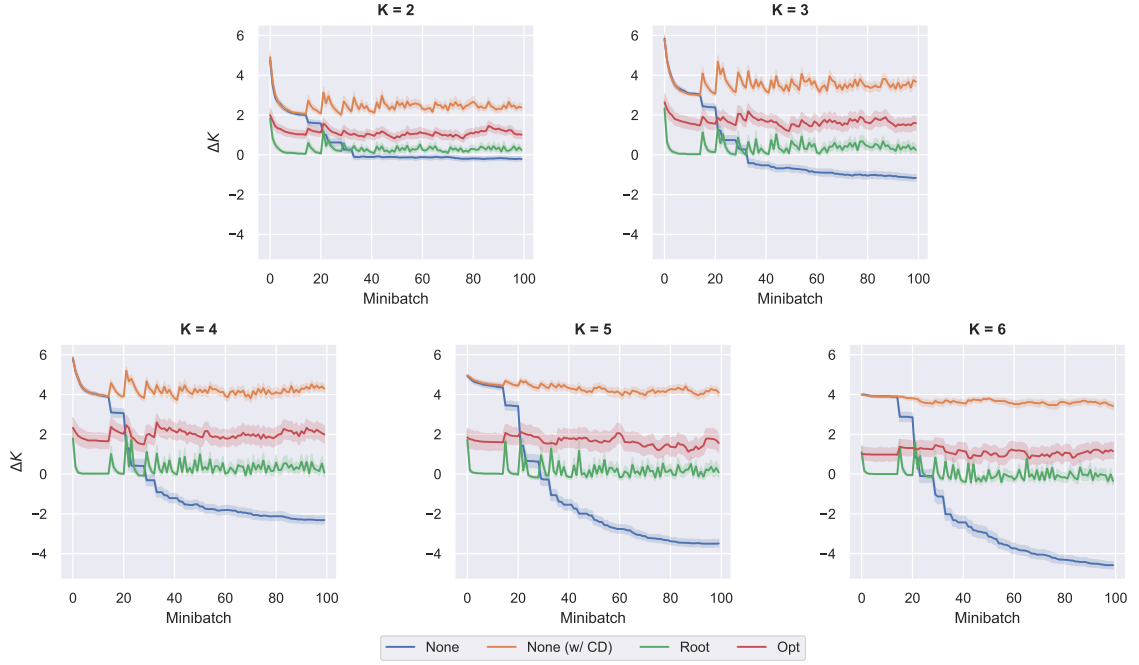


Figure 5.6: Average per-minibatch ΔK for each of the algorithm configurations given in Table 5.2 when tested on the radial dataset with $K = 2, \dots, 6$.

5.4 Summary

In this chapter, we demonstrated the effectiveness of our algorithm by comparing its performance to the baseline online BNP clustering as well as online BNP clustering with change detection. Our experiments indicate that the proposed algorithm outperforms the baseline in terms of cluster label accuracy (measured by AMI and KL) as well as estimation of the number of clusters in the data (measured by ΔK). Proper selection of the hyperparameter and an effective change detection scheme are critical. Both of our hyperparameter tuning methods reduce the over-clustering phenomenon, exhibited by the ΔK for initial minibatches in Figs. 5.3 and 5.6. In general, the root-finding approach finds clustering solutions that better match the generative model. This may be due to the fact that it explicitly bounds the amount of error that can be incurred by increasing r , allowing the tuning algorithm to pick more aggressive hyperparameter values. On the other hand, the optimization-based approach does not require any pre-defined constants, making it easier to use. The root-

finding approach tends to be much faster than the optimization approach. Different choices for numerical solvers (and their parameters) could affect the rate of convergence to a solution for either approach. Change detection accounts for a small amount of the increase in average execution time, but it is obvious that the proposed tuning methods are computationally expensive.

Additionally, we note that the baseline does not adapt well to a dynamically changing number of clusters. Instead of adding new GMM components when the data exhibits new clusters, the baseline grows existing components to envelop the new data. This means that while clusters may be deleted when necessary, they are rarely added. Change detection is effective in curbing this tendency, but over-clustering is not avoided without hyperparameter tuning. It is our novel combination of change detection and hyperparameter tuning for the PCRPP that enables the proposed algorithm to outperform the baseline.

Chapter 6

Experiments with radar pulse data

Dynamic spectrum access (DSA) enables scarce spectrum resources to be shared between licensed primary users and opportunistic secondary users (PUs and SUs, respectively), as long as SUs do not interfere with the operation of the PUs [101]. DSA can help relieve spectrum congestion when there are many transmitters in the same local area. To achieve this, SUs must be able to sense for holes in the spectrum. When the PUs are mainly pulsed radars, it can be necessary for cognitive SUs to identify and cluster pulses in order to predict when spectrum holes may appear or to provide information about PUs to other SUs [102], [103]. Other radar-centric applications of DSA include spectrum coexistence of radar and communications systems [104], [105] and spectrum sharing between automotive radars [106].

Cognitive SUs must operate efficiently to process the large number of pulses they can encounter in a congested spectrum. The processing chain starts with a detector that extracts features such as radio frequency (RF), pulse width (PW), and angle of arrival (AOA) from the pulses in the signal. These features are sometimes referred to as pulse descriptor words (PDWs). Detection results in a sequence of pulses from multiple emitters that are interleaved in time. Depending on the application, a deinterleaving step may be necessary to disambiguate the signals.

Numerous radar pulse deinterleaving algorithms have been described in the literature, e.g., [17], [107]–[112]. Data clustering is a key component in several of these approaches. Other concepts in cognitive radio may also rely on radar pulse clustering, e.g. to help classify emitters in order to predict spectrum utilization [113]. The large variety of existing clustering approaches have different properties, some of which are more ideal for DSA radar applications. For example, in a truly dynamic spectrum environment, the number of emitters at a given time is not known, leading to an unknown number of clusters that can change over time. Therefore, clustering algorithms used for this purpose should not rely on a priori knowledge of the number of clusters. Additionally, because radars may produce large numbers of pulses in a short period of time, it is important that the clustering algorithm is able to process data efficiently.

Online implementations of BNP methods, in particular, have been shown to be effective for clustering radar pulses [38]. However, we have already shown that standard BNP methods can over-estimate the number of clusters. In this chapter, we apply our algorithm to the radar pulse clustering problem and show that it can provide better estimates on the number of emitters in three scenarios. The scenarios are presented in increasing order of pulse density and emitter count. A moving platform with limited detection range travels through the scenario and collects pulses into minibatches. Each minibatch is clustered according to the probabilistic model described in Chapter 3, resulting in a cluster assignment for each pulse and an estimate of the number of emitters. The number of emitters within detection range changes over the course of the scenario. We show that, in these scenarios, the classical CRP-based approach fails to provide good clustering solutions unless change detection is implemented to signal the addition of new emitters. When change detection is added, the classical approach then over-estimates the number of emitters. As the number of emitters increases, the recently proposed online variational PCRP-based methods provide better estimates while maintaining good clustering solutions.

This chapter extends [38] to include an expanded algorithm set with simulations across

a wider variety of conditions. Although [38] appears to be the first work applying BNP to radar pulse clustering, the work presented here demonstrates that additional innovation is necessary to achieve robust performance. We show that incorporation of the novel online PCR-P-based clustering algorithm proposed in Chapter 3 can provide this robustness. This is the first work demonstrating our algorithm on radar pulse data.

6.1 Related work

Numerous techniques for clustering radar pulses have been described in the literature. We provide an overview of recent related work and emphasize the uniqueness of our approach. An adaptively-weighted fuzzy C -means approach was proposed by [114]. Fuzzy C -means also serves as a basis for the method in [109], but the approach requires multiple sensors. Our work employs only a single sensor to cluster signals from multiple emitters. Unlike our work, the fuzzy C -means based methods require the number of clusters to be specified a priori and do not operate in an online paradigm for streaming radar pulse data. The support vector clustering approaches in [115], [116] do not require the number of clusters to be known beforehand, but still do not operate online.

Self organizing maps and fuzzy adaptive resonance theory have also shown promise for clustering radar pulses [111], [117]. These algorithms are not probabilistic, however, and are not designed for datasets that have a dynamically changing number of clusters. As we will show in this chapter, our approach can adapt online when the number of emitters changes over time while maintaining a fully probabilistic inference framework.

Other approaches rely on heuristic distance thresholding to determine cluster membership [17], [118]–[120]. These approaches are highly optimized for online processing, but are not probabilistic. Cluster membership thresholds are thus chosen in an ad-hoc or heuristic manner. Our fully Bayesian approach, on the other hand, can capture the uncertainty inherent in radar pulse clustering and utilizes a probability model to determine

cluster assignments instead of distance in feature space.

Finally, studies on variational Gaussian mixture models for radar pulse clustering are perhaps most similar to our work [38], [121], [122]. Although the approaches in [121], [122] are able to estimate the number of clusters in the data, this is achieved by evaluating the ELBO for different values of $K \in [K_{\min}, K_{\max}]$ and choosing the K that yields the highest ELBO. Moreover, these works do not consider the online processing paradigm, nor do they account for changes in K mid-stream. In this work we show that K can be estimated jointly with the cluster parameters online by incorporating a BNP prior like in [38]. But that work relied solely on classical CRP-based methods for a single radar scenario. We expand the number of scenarios and compare the classical approach with the improved PCR-P-based method proposed in Chapter 3.

6.2 Simulation

We now present three radar pulse clustering scenarios with increasing pulse density. These experiments demonstrate that a BNP clustering algorithm that is resistant to over-clustering can provide better solutions than the standard CRP-based approach. The first scenario is based on the one presented in [38] with the addition of a moving platform with limited detection range and emitter placement in a Cartesian coordinate system. In each scenario, the platform traverses a linear flight path from $(-60, -60)$ m to $(60, 60)$ m in 10 seconds at a constant speed (approximately 17 meters per second). This corresponds to the speed capability of many consumer-grade drones. The modeled platform has a detection range of 80 meters, which means that the number of detected emitters at a given time may be less than the total number of emitters in the scenario. When an emitter is within the detection range, the probability of detecting a pulse is $P_D = 0.98$. Over the course of the scenario, the platform collects pulses in 1000 minibatches of equal duration. Note, however, that the number of pulses in each minibatch may differ due to the probability of detection, the

number of emitters within detection range, and the PRFs of those emitters.

There are six emitters in Scenario 1. The second scenario adds three emitters to the first scenario which increases the number of clusters in the data space to nine. Similarly, the third scenario builds upon the first two by including three additional emitters—twelve in all—at a higher average PRF than the emitters in Scenarios 1 and 2. This results in increased per-cluster pulse density which can impact the ability of the standard CRP-based clustering approaches to accurately estimate the number of clusters. This is because the expected number of clusters in the standard CRP grows logarithmically with the number of data points [72].

Table 6.1 lists the emitters considered in each scenario along with their parameters. Fig. 6.1 provides a layout for the emitters in each scenario. Note that the scenarios are cumulative—each scenario includes the emitters from the prior scenarios in addition to new emitters. The platform flight path is indicated by a broken line. The color of the emitter marker represents the emitter’s PRF. Fig. 6.2 shows the RF and PW feature cluster centers and second standard deviations in each dimension. The legend follows the same convention as Fig. 6.1. Clustering algorithms use RF and PW as features but do not utilize PRF—the PRF only impacts the number of points in a cluster for a given minibatch. The angle of arrival (AOA), measured in degrees, is used as a third feature for clustering. Additive Gaussian noise, distributed with zero mean and unit variance, was applied to the AOA, corresponding to the Cramer-Rao lower bound for AOA estimates at roughly 5 dB SNR [123]. The AOA changes for each emitter as the platform moves along its linear trajectory, which means that the emitter clusters move throughout feature space. Online clustering is able to efficiently update the cluster centers according to changes in the AOA feature. Figs. 6.3 and 6.4 depict the time-varying number of clusters within detection range and detected pulses for each scenario. The number of clusters changes over time because as the platform moves through the scene, the number of emitters within its detection range changes.

Table 6.1: Emitter descriptions for the simulated scenario. Frequency (GHz) and pulse width (μs) entries are given as $[\mu, \sigma]$ pairs corresponding to independent draws from a Gaussian distribution with mean μ and variance σ^2 . PRF is given in kHz . Emitter time on and time off are given by t_{on} and t_{off} , respectively.

Frequency (GHz)	Pulse width (μs)	PRF (kHz)	(x, y) (m)
Scenario 1			
[1, 0.1]	[2, 0.1]	10	(−50, 50)
[2, 0.1]	[5, 0.1]	8	(0, 50)
[5, 0.1]	[1, 0.1]	5	(−40, −20)
[3.75, 0.25]	[5, 0.1]	20	(50, −30)
[5, 0.1]	[5, 0.3]	8	(30, 0)
[6, 0.1]	[2, 0.1]	10	(0, −20)
Scenario 2 (in addition to Scenario 1)			
[1, 0.1]	[4, 0.25]	5	(30, 30)
[2, 0.25]	[3, 0.25]	10	(−50, 30)
[4, 0.1]	[3, 0.1]	6	(0, 0)
Scenario 3 (in addition to Scenarios 1 & 2)			
[6, 0.1]	[4, 0.1]	40	(−50, −40)
[2, 0.1]	[1, 0.25]	15	(−20, 20)
[4, 0.3]	[4, 0.3]	25	(10, 10)

Note that the emitter RF and PW features studied here are notional. The algorithms employed in this work do not rely on specific feature values. In principle, these algorithms can be used to cluster data from emitters in any band, transmitting with any pulse width. However, the overall pulse density in a given application, driven by the emitters' PRFs, obviously has an impact on the computation time. If the pulse density is very high, smaller minibatches may be necessary. Management of data rates is primarily a function of the available processing hardware on the platform and is not considered in this study. Moreover, other aspects of the simulation including the spatial separation of emitters and platform speed are provided for illustrative purposes and do not have direct impact on the clustering algorithms being studied.

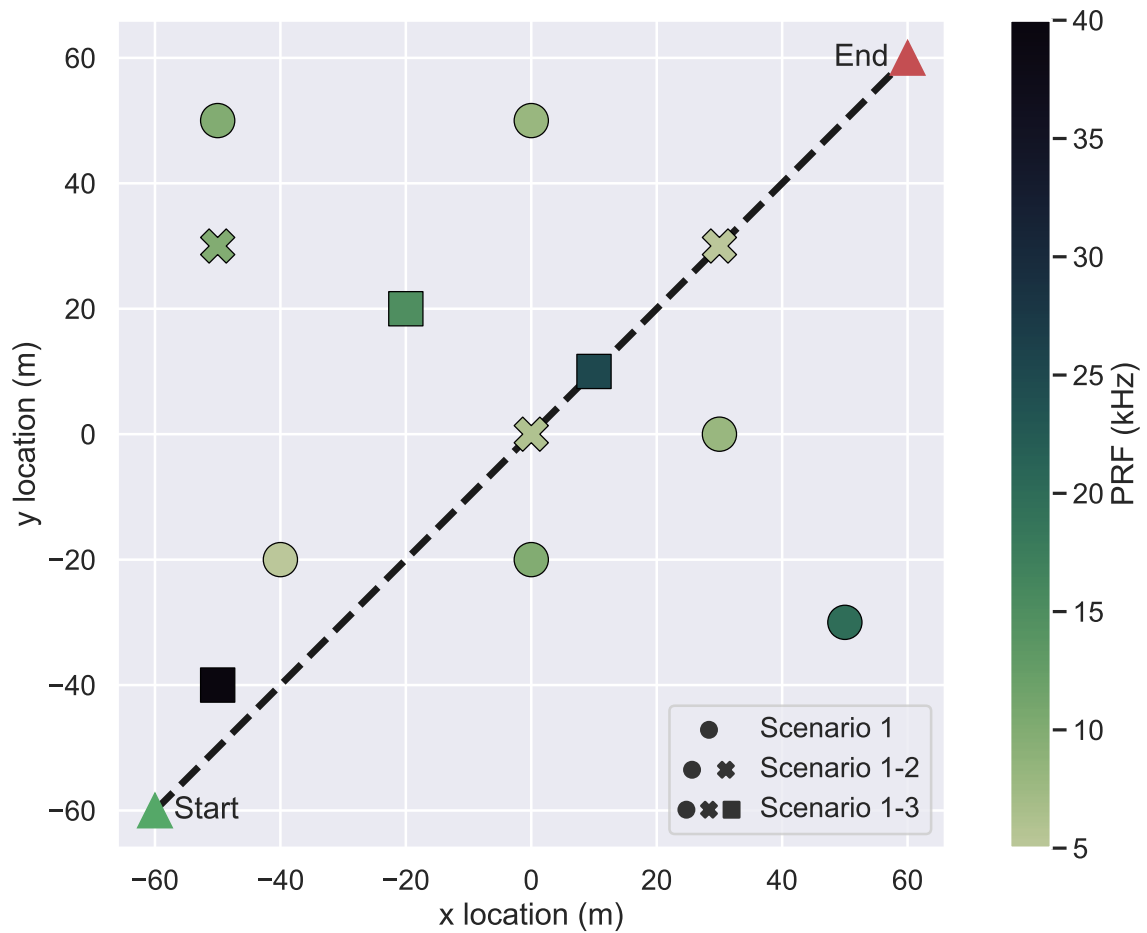


Figure 6.1: Layout for each of the three scenarios studied in this work. Note that the symbols indicated by the legend are cumulative across scenarios, i.e., each scenario includes all of the emitters in the previous scenarios in addition to three more. The platform flight path is indicated by a broken line and is kept consistent for each scenario. The color of the marker indicates the emitter PRF which is *not* used as a feature for clustering.

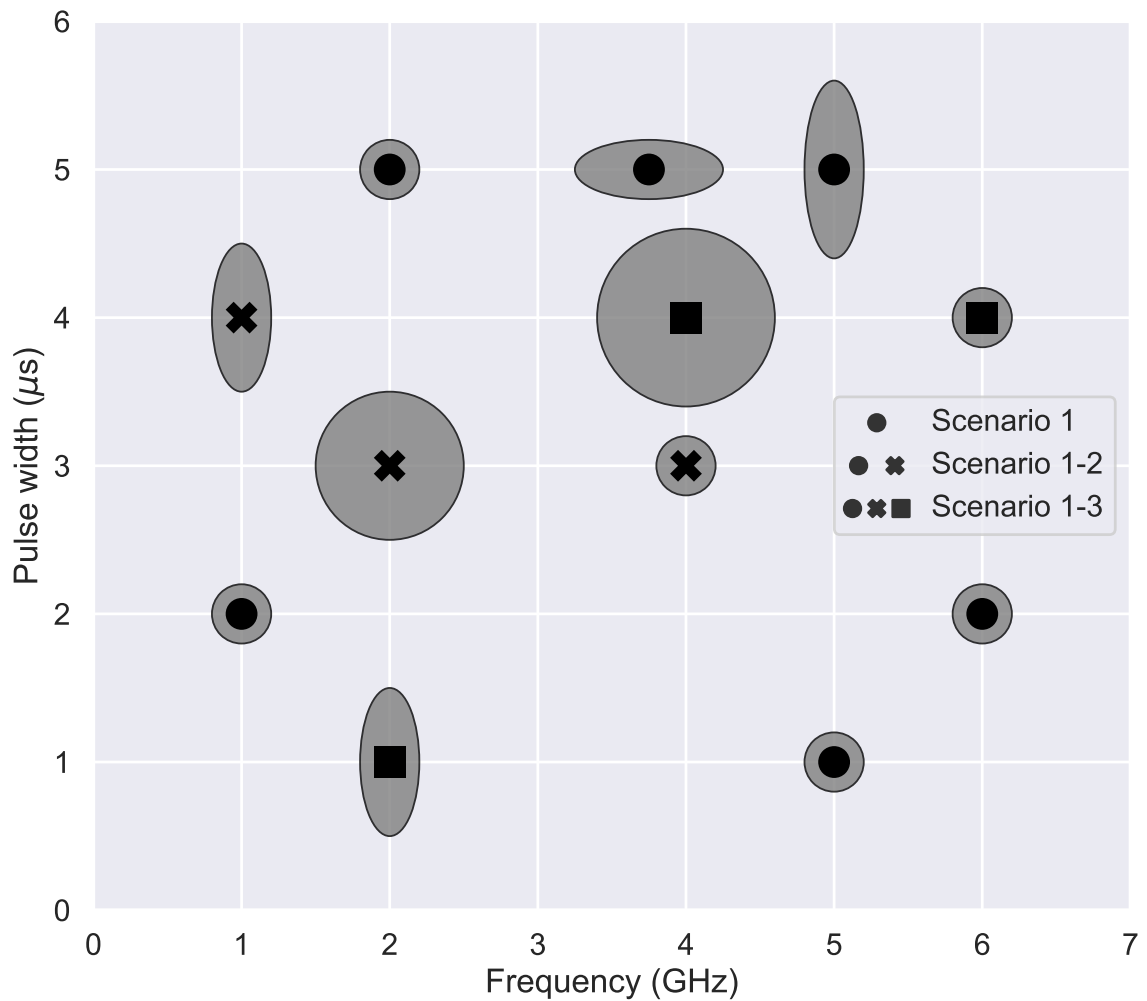


Figure 6.2: Pulse cluster shapes for each of the three scenarios studied in this work. Note that the symbols indicated by the legend are cumulative across scenarios, i.e., each scenario includes all of the emitters in the previous scenarios in addition to three more. The ellipses indicate the second standard deviation of the pulses in that cluster.

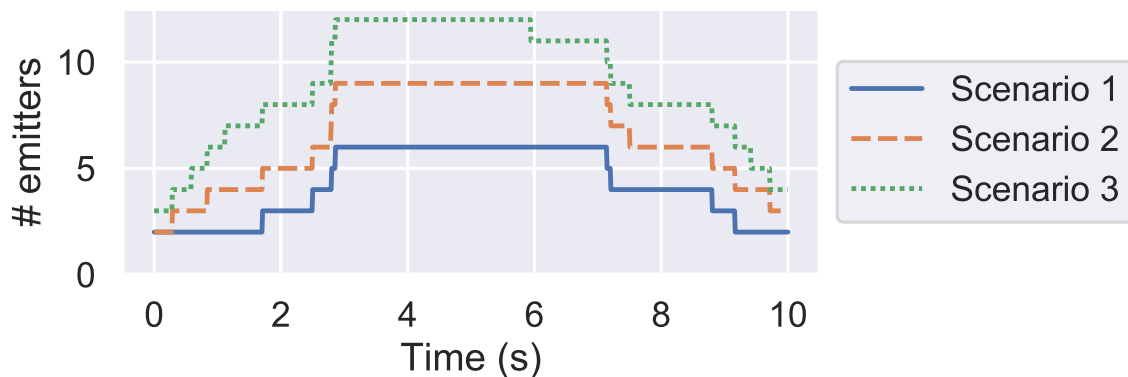


Figure 6.3: Number of emitters within detection range as a function of time for each scenario. Each time step corresponds to a minibatch. Note that the number of emitters is equal to the number of clusters in the data at each time step. The maximum number of simultaneous clusters increases from Scenario 1 to Scenario 2 and 3.

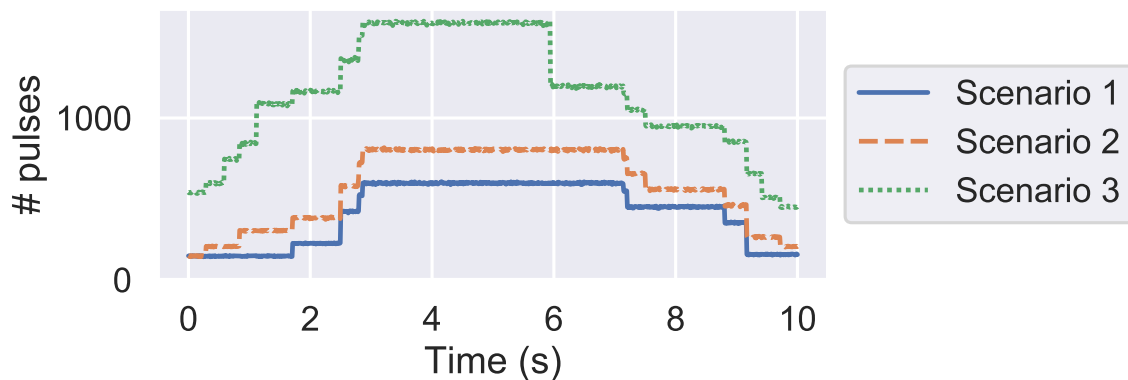


Figure 6.4: Number of detected pulses as a function of time for each scenario. Each time step corresponds to a minibatch. The maximum number of detected pulses in a minibatch increases from Scenario 1 to Scenario 2 and 3. Scenario 3 has a considerably higher pulse count because the emitters added for this scenario have relatively higher PRFs.

Table 6.2: Prior hyperparameters for the Normal-Wishart data model in radar pulse clustering tasks. A $D \times D$ identity matrix is denoted in this table by $\mathbf{I}$, where D is the data dimensionality. A vector of D zeros is denoted by $\mathbf{0}$.

Hyperparameter	Value
$\mathbf{m}_0$	$\mathbf{0}$
ν_0	0.01
$\mathbf{W}_0$	$\mathbf{I}$
λ_0	D

6.2.1 Methodology and metrics

We utilize the algorithm configurations and metrics outlined in Section 5.1 for the analysis presented in this chapter. In particular, we use the algorithm configurations and shorthand naming conventions listed in Table 5.2. Table 6.2 lists the prior hyperparameters for the Normal-Wishart data model. The precision matrix is constrained to be diagonal. For these experiments, we set $K_0 = 15$ for the initial cluster count and $\eta = 0.06$ for the root-finding tuning method. We report the AMI and ΔK —for a discussion on algorithm timing, see Chapter 5.

6.3 Results

Scenario 1 features up to six simultaneous emitters at various PRFs. Fig. 6.5 depicts the aggregated AMI and ΔK metrics for the entire scenario. The baseline classical approach (tuning method “None”) performs poorly relative to the rest of the tested approaches. The large negative ΔK achieved by the baseline approach can be diagnosed by examining the per-minibatch Monte-Carlo average ΔK plotted in Fig. 6.6. This confirms the trend seen in Chapter 5, where the baseline approach fails to add new mixture components when the data exhibits new clusters. Incorporating change detection, like the SPLL approach employed here (tuning method “None (w/ CD)”), resolves this tendency, but brings with it the documented tendency for BNP clustering algorithms to over-cluster [27], [29]. Tuning the

PCR-P hyperparameter online using root-finding, in conjunction with SPL-L change detection, achieves high AMI and ΔK near zero for most minibatches. The optimization-based tuning method fails to provide good estimates of the number of clusters and consistently over-estimates this quantity like the untuned method with change detection (“None (w/ CD)”). This may be due in part to the fact that N , the number of data points in a minibatch, is approaching the maximum value that is modeled by $\hat{p}(K \mid r)$ in (4.5), thereby potentially limiting its applicability.

Scenario 2 introduces three emitters in the same PRF range as the emitters in Scenario 1. As before, Fig. 6.7 depicts the overall average AMI and ΔK for each algorithm configuration. The baseline approach (tuning method “None”) performs poorly relative to the other three approaches. Introducing change detection (tuning method “None (w/ CD)”) again causes an increase in average AMI, but does not resolve overclustering. The tuned PCR-P-based online clustering algorithms achieve superior AMI and ΔK , with the root-finding method again providing the best result. Fig. 6.8 shows that the baseline approach essentially collapses down to a single cluster after three seconds and never recovers from that point, while the other methods are able to keep up with changes in the data stream.

Finally, Scenario 3 adds another three emitters with higher PRFs than the emitters in Scenario 1 and 2. Fig. 6.9 shows that there is a more significant impact on ΔK for the baseline method with change detection (tuning method “None (w/ CD)”), but the tuned methods produce clustering solutions with a better estimation of the number of clusters. On average, the root-finding method is best overall. However, ΔK for the root-finding method does dip below zero after about 7 seconds as shown in Fig. 6.10, indicating a failure to add new mixture components. At the same time, the optimization-based method estimates the correct number of clusters, while the standard approach with change detection over-estimates it. The baseline approach suffers from the same cluster “collapse” as it did in

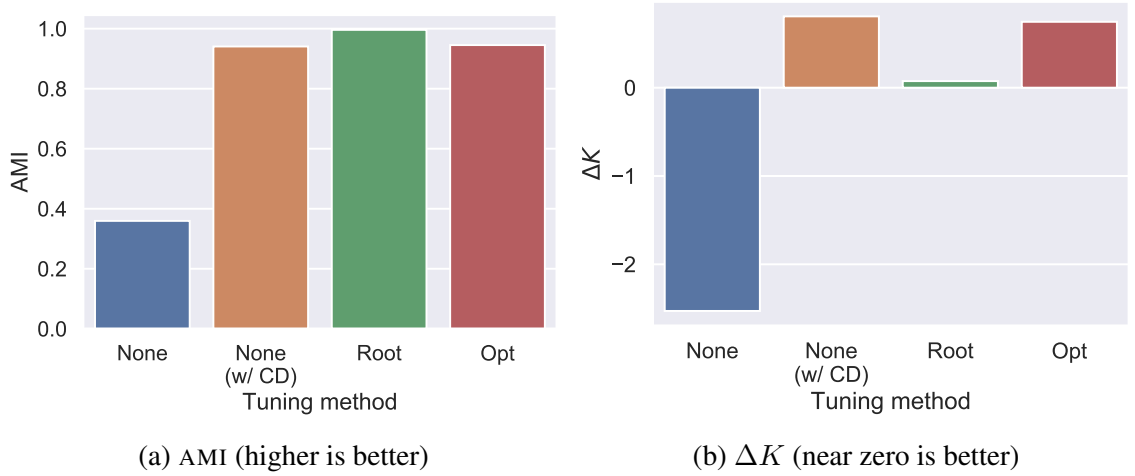


Figure 6.5: Performance metrics for each of the algorithm configurations given in Table 5.2 when tested on Scenario 1. Metrics were computed for each minibatch, averaged across all Monte-Carlo experiments, and then averaged across minibatches.

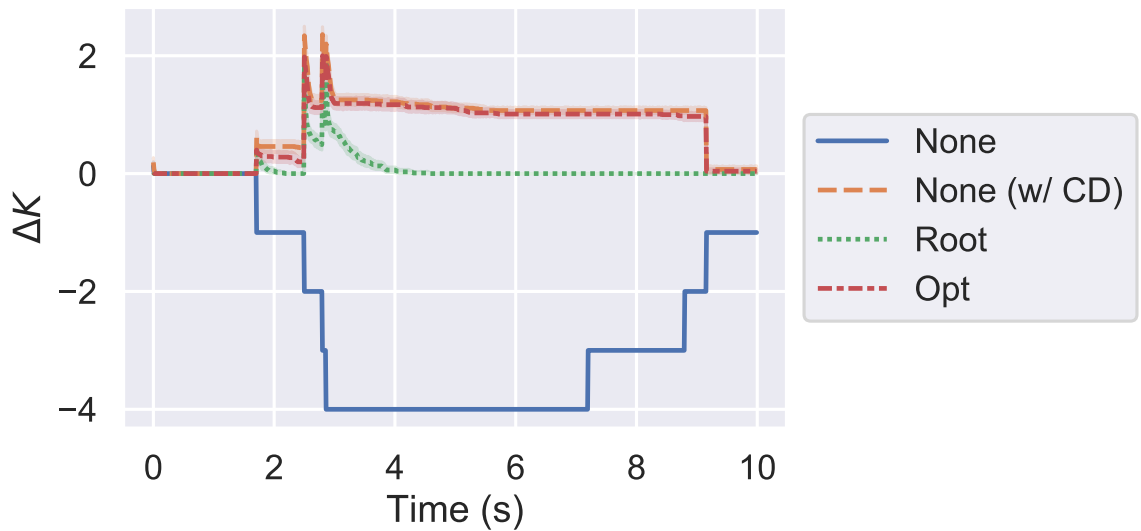


Figure 6.6: Average per-minibatch ΔK for each of the algorithm configurations given in Table 5.2 when tested on Scenario 1.

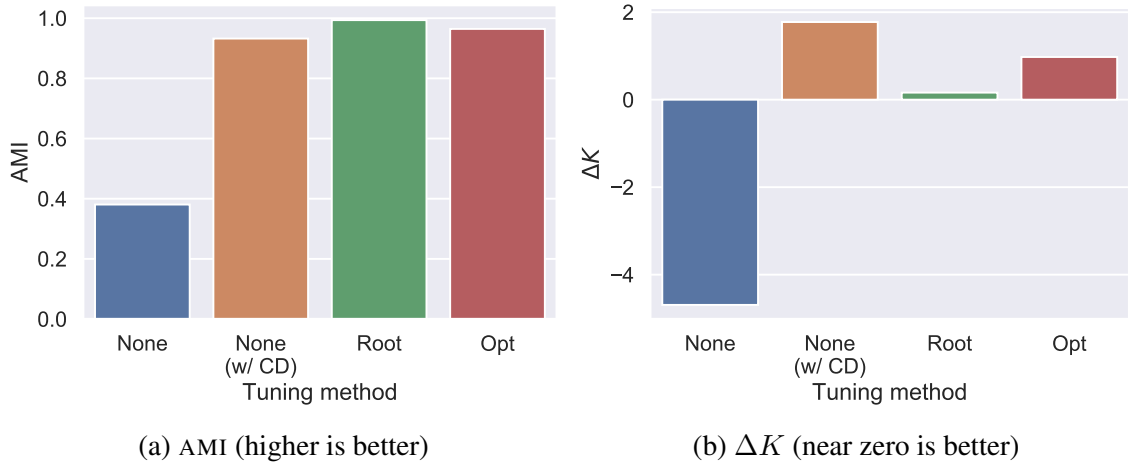


Figure 6.7: Performance metrics for each of the algorithm configurations given in Table 5.2 when tested on Scenario 2. Metrics were computed for each minibatch, averaged across all Monte-Carlo experiments, and then averaged across minibatches.

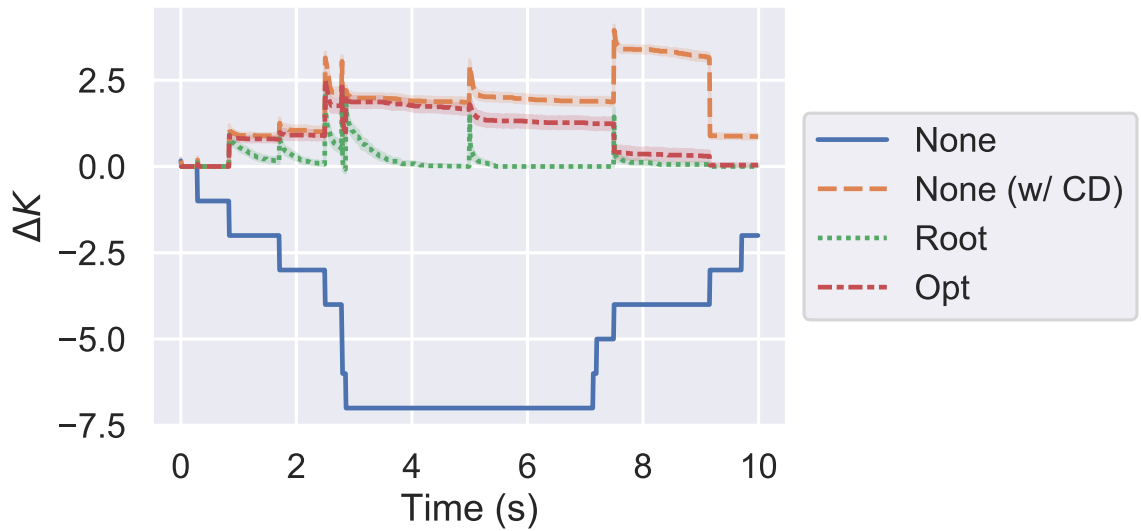


Figure 6.8: Average per-minibatch ΔK for each of the algorithm configurations given in Table 5.2 when tested on Scenario 2.

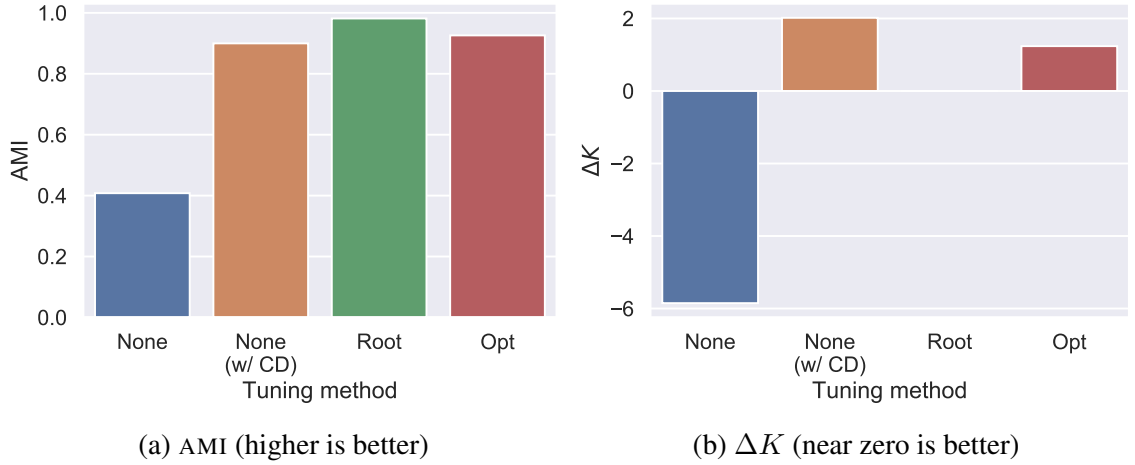


Figure 6.9: Performance metrics for each of the algorithm configurations given in Table 5.2 when tested on Scenario 3. Metrics were computed for each minibatch, averaged across all Monte-Carlo experiments, and then averaged across minibatches.

Scenarios 1 and 2.

6.4 Summary

In this chapter, we employed the Bayesian nonparametric clustering algorithm proposed in this dissertation to cluster radar pulses in a dynamically changing spectrum environment. We evaluated four algorithm configurations on three radar pulse clustering scenarios of increasing density. For each scenario, PCR-P-based methods yielded the best results. The root-finding hyperparameter tuning method consistently provided good estimates on the number of emitters, while the classical CRP-approaches underestimated it. When change detection was added to signal incoming changes in the data statistics, the classical CRP-based approach then overestimated the number of emitters. This shows that when considering BNP clustering methods for radar pulse data, it is important to select appropriate priors. We note that although optimization-based tuning can reduce over-clustering without requiring user-specified hyper-parameters (like η in the root-finding method), its performance is not as good as the root-finding method.

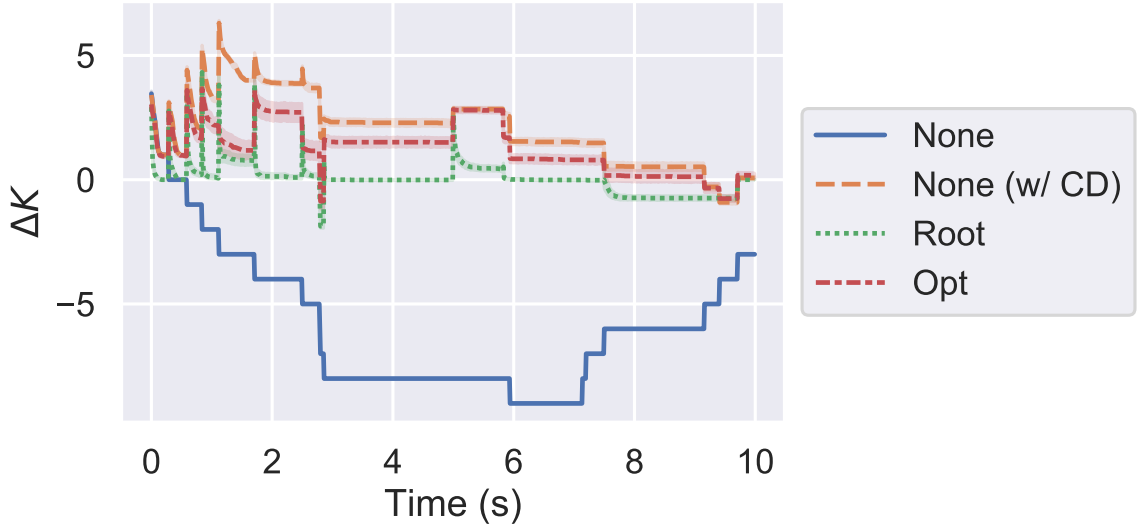


Figure 6.10: Average per-minibatch ΔK for each of the algorithm configurations given in Table 5.2 when tested on Scenario 3.

This chapter provides new options for clustering radar pulses in dynamic spectrum access applications. Clustering can be useful when predicting temporal availability in the spectrum [113] or for processes that develop radio environment maps [103]. Many deinterleaving algorithms utilize a clustering step [17], [107]–[112]—the algorithms studied here could be incorporated into a fully Bayesian deinterleaver that characterizes the uncertainty in a deinterleaving solution. We also note that the temporal characteristics of radar signals are ignored in this work. Clusters are assumed to contain i.i.d. samples from a multivariate Gaussian distribution, but this is not necessarily correct. Future work could focus on a better data model that accounts for these discrepancies.

Chapter 7

Neural spike sorting with BNP modeling

Spike sorting is a neurobiological application in which neuron voltage spikes are clustered based on their shapes [124]. Because the signal spikes corresponding to individual neurons are unique, the clusters found by a spike sorting algorithm can be used to identify different neurons. These spikes are usually recorded with thin electrodes implanted in the brain. A single electrode records the activity of multiple neurons, which means that the spikes in the recording need to be detected and separated in order to associate them to the correct neuron. Activity of specific neurons (or groups of neurons) may reflect a response to a particular treatment or may be used for characterizing memory processes [125]. Spike sorting is an important step in associating neural activities.

Signals recorded by the electrodes are generally filtered first before spike detection. Features are extracted from detected spikes and clustered. The cluster results are then associated with the original detected spikes. The number of neurons (clusters) is usually unknown beforehand, so spike sorting algorithms must somehow infer the number of neurons. Many clustering algorithms have been applied to spike sorting, including BNP clustering [126]–[128] and online clustering [37], [129]–[132], but the spike sorting literature is still active. Some techniques intentionally over-cluster data in order to discover the correct number of clusters [133]. To our knowledge, ours is the first work combining Bayesian

nonparametrics and online variational processing for spike sorting.

7.1 Experiments

To demonstrate the proposed algorithm for spike sorting, we evaluate two commonly studied artificial datasets, which we refer to as *quiroya*¹ [135] and *wild*² [134]. These datasets are ideal for this work because of the availability of ground truth and benchmark performance reported in [134] for comparison. Each dataset contains multiple subsets at different noise levels. We evaluate data subsets with noise standard deviation ranging from 0.05 to 0.15 (see [134] for more details).

As is common practice when evaluating clustering algorithms for spike sorting (e.g., [134]), ground truth spike locations are used for extracting spikes instead of relying on a spike detector—this way, algorithm performance is not impacted by detector errors. Waveform features are extracted via the weighted principle components analysis (PCA) [19] of wavelet features approach utilized in [133] and described here in brief.

7.1.1 Spike feature extraction via weighted PCA of wavelets

The wavelet decomposition of a detected spike $s[n]$, where n is the sample index, is given by

$$c_{j,k} = \sum_n s[n] \psi_{j,k}[n] \quad (7.1)$$

with

$$\psi_{j,k}[n] = \frac{1}{\sqrt{2^j}} \psi \left(\frac{n - k2^j}{2^j} \right) \quad (7.2)$$

where $c_{j,k}$ is the wavelet coefficient corresponding to the j -th scaling and k -th translation factor and ψ is the mother wavelet. This approach utilizes the Haar wavelet [136].

¹titled “quiroya-easy-1” in [134]

²titled “JW short” [134]

R replicates of a GMM with K clusters are used to cluster the z-scores [137] of the wavelet coefficients for the collection of spikes. The replicate with the highest log likelihood is kept as a way to diminish the impact of random initial conditions on GMM fitment. The number of replicates R and clusters K is configurable. Note that these GMMs are used for feature selection and are non-Bayesian and un-collapsed, meaning that the number of clusters is fixed and the cluster weights w_k are also computed. To reduce confusion between these GMMs used for feature selection and the BNP GMM developed in this dissertation, we refer to the former as the “FeatGMM” to indicate that it is used for feature extraction.

The FeatGMM is used to derive a scaling coefficient before extracting features in the following way. First, compute the pairwise distances between the i -th and j -th FeatGMM components, defined by

$$d_{i,j} = \frac{|\mu_i - \mu_j| \sqrt{w_i - w_j}}{\sqrt{\sigma_i^2 - \sigma_j^2}}. \quad (7.3)$$

Then, multiply the z-scored wavelet coefficients by the median of these distances and compute the PCA from these scaled coefficients. The first D PCA coefficients are used for clustering. For more details, see [133].

7.1.2 Results

We demonstrate our algorithm for clustering spike features extracted by the method in [133]. We report results for both of the tuning methods described in Chapter 4: the optimization-based method (“Opt”) and the root-finding method (“Root”) with $\eta = 0.06$. Our algorithm was configured with $K_0 = 10$. Table 7.1 lists the prior hyperparameters used for the Normal-Wishart data model. We do not constrain the precision matrix to be diagonal in this case because there is no evidence that the extracted feature space exhibits this characteristic. For the FeatGMM, we used $R = 4$ replicates with $K = 4$ and clustered the first $D = 4$ PCA coefficients.

The *quirolga* dataset has three subsets corresponding to noise standard deviations 0.05,

Table 7.1: Prior hyperparameters used for the Normal-Wishart data model in spike sorting tasks. A $D \times D$ identity matrix is denoted in this table by $\mathbf{I}$, where D is the data dimensionality. A vector of D zeros is denoted by $\mathbf{0}$.

Hyperparameter	Value
$\mathbf{m}_0$	$\mathbf{0}$
ν_0	0.1
$\mathbf{W}_0$	$\mathbf{I}$
λ_0	D

0.1, and 0.15. Each subset contains three neurons and an average of 1738 spikes (datapoints) each. The *wild* dataset is subdivided by noise level (standard deviations 0.1 and 0.15) and neuron count (ranging from one to nine neurons), for a total of 18 data subsets with an average of 419 spikes (datapoints) each. We processed the data subsets individually by dividing each subset into minibatches of size $\min(300, |\mathcal{X}|)$. After the final minibatch of a data subset is processed by our online BNP clustering algorithm, a pass over the whole data subset is made to make a final assignment of each datapoint in the data subset to its nearest cluster. Cluster parameters are not updated during this step. AMI is computed from ground truth for each data subset and averaged.

We note that although the number of neurons in a data subset theoretically corresponds to the number of clusters, this is not necessarily the case because the feature extraction method may fail to create separate clusters for each neuron or erroneously split neurons among multiple clusters. Because we cannot visualize $D = 4$ dimensions, it is difficult to verify that the number of neurons in the dataset corresponds to the number of clusters in the extracted features. This serves as motivation for data driven algorithms like the one proposed in this dissertation that can automatically infer the number of clusters in the data. On the other hand, we are unable to compute ΔK , the error in the estimated number of clusters, because the true number of clusters is unknown.

We compare performance to the results reported in [134] for the KlustaKwik algorithm [138] as a baseline. We chose KlustaKwik as a baseline because like our algorithm, it

Table 7.2: Algorithm performance and baseline comparison, measured by AMI. Higher is better. The best score for each dataset is indicated by **boldface text**. The columns labeled “Root” and “Opt” are results for our algorithm with the root-finding and optimization-based tuning methods, respectively. The “Baseline” column corresponds to results for KlustaKwik [138], a GMM-based spike sorting method. The column labeled “Best” gives the best reported results in [134] for the dataset, obtained by the WaveClus algorithm [135]. For both datasets, the baseline and best performance was obtained from the supplementary material for [134], retrieved on March 1, 2020.

Dataset	Our approach		Reported in [134]	
	Root	Opt	Baseline	Best
<i>quiroga</i> [135]	0.941	0.894	0.863	0.910
<i>wild</i> [134]	0.413	0.586	0.542	0.562

implements a GMM and automatically infers the number of clusters, but under a different inference framework. We also include the performance of the WaveClus algorithm [135], the best overall performing algorithm identified in [134].

Table 7.2 lists the AMI for the selected datasets, averaged across all noise levels and neuron counts. Our results indicate that the proposed algorithm outperforms the KlustaKwik baseline on *quiroga* and *wild*. Additionally, when root-finding tuning is used on *quiroga* and optimization-based tuning is used on *wild*, our algorithm even outperforms WaveClus, the best performing approach from [134]. We note that the AMI for *wild* is relatively low for all of the clustering algorithms reported in [134], indicating that this dataset is more challenging than *quiroga*. Interestingly, we also note that the optimization-based tuning method yields superior results to the root-finding method on *wild*, counter to the results we found with other datasets described in this dissertation.

Fig. 7.1 depicts the average spike shapes that were extracted from example datasets from *quiroga* and *wild*. For both examples, the noise in the data has standard deviation 0.1. The results in the figure were obtained using optimization-based tuning. Fig. 7.1a clearly depicts three distinct spike shapes, which agrees with the ground truth. Fig. 7.1b shows that our method estimates one cluster too many (the data truly contains four clusters), but

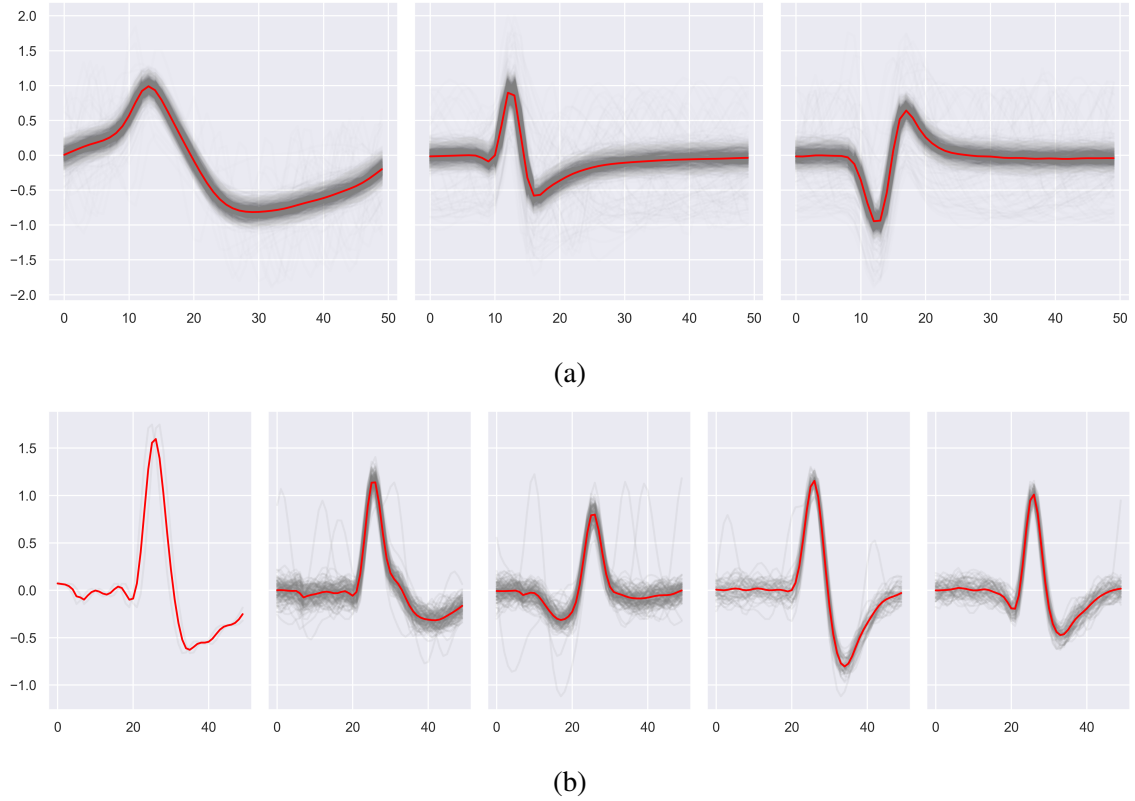


Figure 7.1: Clustered spike shapes from the (a) *quiroga* and (b) *wild* dataset with noise standard deviation 0.1, extracted by our proposed algorithm. The spikes grouped together in each cluster are plotted in gray with transparency. The average of the spikes for each cluster is plotted in red.

the extra cluster only contains a few data points.

7.2 Summary

The experiments carried out here show that our algorithm is a promising approach for GMM-based spike sorting. We demonstrated this on two artificial spike sorting datasets that have been used in other works to compare existing spike sorting solutions [134]. Our algorithm was found to be competitive with other methods that are specifically designed for spike sorting for processing selected datasets from [134], [135]. Note that our proposed algorithm clusters the data online, whereas the competing methods operate in the offline

paradigm. Because of this, our algorithm could be employed for identifying spikes in large datasets. The online paradigm could also be appropriate for real-time spike sorting [37]. Further investigation on a larger set of neural spike data should be pursued in order to establish a better performance characterization and to fine-tune the performance of our algorithm.

Chapter 8

Clustering data with mixed type

Although GMMs can be effective for clustering real-world data despite the potential for non-Gaussianity, many datasets have features with mixed Gaussian and categorical type. Mixed datasets appear frequently in numerous fields, including medicine, economics, and marketing [139]. Encoding of categorical information must be treated carefully. A common approach is to use numerical encoding for categorical information. This may not result in quality clustering, however. For example, consider encoding two categories as $A \rightarrow 1$, $B \rightarrow 2$. Treating these categories as numbers is problematic because it is possible to obtain a numerical cluster that is “between” categories A and B , corresponding to a numerical value between 1 and 2. But this numerical value does not correspond to a category, so interpretation of such a result is difficult. Moreover, when there are more variables to be clustered, permuting the mapping (i.e., $A \rightarrow 2$ and $B \rightarrow 1$) could result in a change in the clustering result, even though the underlying (categorical) data is the same. This motivates a more thoughtful approach for dealing with categorical data and mixed data in general.

In this chapter, we extend our online BNP clustering algorithm to handle variables of mixed Gaussian and categorical types. We also extend the NISE (originally defined for GMMs and discussed in Chapter 4) to mixed data models. These contributions provide new methods for handling data with mixed types. To our knowledge, this is the first time a BNP

model of this kind—one with explicit control over redundancy in the mixture—has been proposed for mixed data clustering. BNP has been used for mixed data before (e.g., [140], [141]), but none of the prior clustering work utilizes variational inference or considers the problem of redundancy or over-clustering. We demonstrate the proposed mixed-data mixture model on two datasets with a qualitative data mining interpretation.

8.1 A BNP model for mixed data types

We now formulate a BNP mixture model for data of mixed Gaussian and categorical type. To do this, we append categorical variables to the standard multivariate Gaussian data model. Assuming conditional independence of the multivariate Gaussian and categorical variables, we derive the variational E and M steps by exploiting conjugacy between categorical and Dirichlet random variables. The Dirichlet hyperparameter allows us to control the level of “mixing” categories within one cluster—for instance, when the hyperparameter is above 1, we allow clusters to have points from different categories. When the hyperparameter is below 1, we encourage clusters to only contain one category.

For convenience, partition the n -th feature vector as $\mathbf{x}_n = (\check{\mathbf{x}}_n; \tilde{\mathbf{x}}_n)$, where $\check{\mathbf{x}}_n \in \mathcal{R}^d$ is multivariate Gaussian distributed and $\tilde{\mathbf{x}}_n = (\tilde{x}_{n,1}, \dots, \tilde{x}_{n,C})$ is a vector of C categorical variables with $\tilde{x}_{n,c} \in \{1, \dots, J\}$ for $c = 1, \dots, C$. In other words, each categorical variable $\tilde{x}_{n,c}$ has J levels. Note that we have assumed each categorical variable has the same number of levels to simplify the notation; however, this assumption is made without loss of

generality. Our model is:

$$\theta_k = \mu_k, \mathbf{T}_k, \rho_k \quad (8.1)$$

$$\check{\mathbf{x}}_n \mid \mu_k, \mathbf{T}_k \stackrel{\text{iid}}{\sim} \text{Norm}(\check{\mathbf{x}}_n; \mu_k, \mathbf{T}_k^{-1}) \quad (8.2)$$

$$\mu_k, \mathbf{T}_k \mid \mathbf{m}_0, \nu_0, \mathbf{W}_0, \lambda_0 \sim \text{NormWish}(\mu_k, \mathbf{T}_k; \mathbf{m}_0, \nu_0, \mathbf{W}_0, \lambda_0) \quad (8.3)$$

$$\tilde{x}_{n,c} \mid \rho_{k,c} \stackrel{\text{iid}}{\sim} \text{Cat}(\tilde{x}_{n,c}; \rho_{k,c}), \quad c = 1, \dots, C \quad (8.4)$$

$$\rho_{k,c} \mid \gamma_0 \sim \text{Dir}(\rho_{k,c}; \gamma_0), \quad c = 1, \dots, C \quad (8.5)$$

$$z_n \mid \mathcal{Z}, \alpha, r \sim \text{PCR}(\mathcal{Z}, \alpha, r). \quad (8.6)$$

This model is the same as the GMM derived in Chapter 3 with the addition of the categorical variables (8.4) with parameter ρ and a corresponding Dirichlet conjugate prior (8.5) with hyperparameter γ_0 .

Assume that $\check{\mathbf{x}}_n$ and $\tilde{\mathbf{x}}_n$ are independent given the parameters. Furthermore, assume the categorical features are independent of one another given their respective categorical parameters, i.e.,

$$p(\tilde{\mathbf{x}} \mid \rho_1, \dots, \rho_C) = \prod_{c=1}^C p(\tilde{x}_c \mid \rho_c) \quad (8.7)$$

The original ELBO for a GMM (2.5) then has an additional term for the categorical variables:

$$\begin{aligned} \text{ELBO}(p, q) = & \mathbb{E} \left[\sum_{n=1}^N \log p(\check{\mathbf{x}}_n \mid \mu_{z_n}, \mathbf{T}_{z_n}) \right] + \mathbb{E} \left[\sum_{n=1}^N \sum_{c=1}^C \log p(\tilde{x}_{n,c} \mid \rho_{z_n,c}) \right] \\ & + \mathbb{E} \left[\sum_{k=1}^K \log p(\theta_k) \right] + \mathbb{E} [\log p(\mathcal{Z})] - \mathbb{E} \left[\sum_{k=1}^K \log q(\theta_k) \right] - \mathbb{E} [\log q(\mathcal{Z})]. \end{aligned} \quad (8.8)$$

Variational E and M steps were derived for all of these terms in the fully Gaussian case in

Chapter 2, except for

$$\mathbb{E} \left[\sum_{n=1}^N \sum_{c=1}^C \log p(\tilde{x}_{n,c} \mid \boldsymbol{\rho}_{k,c}) \right] = \sum_{n=1}^N \sum_{c=1}^C \mathbb{E}[\log p(\tilde{x}_{n,c} \mid \boldsymbol{\rho}_{z_n,c})]. \quad (8.9)$$

In the following, we drop the subscript c for clarity and let $z_n = k$ and work with $\mathbb{E}[\log p(\tilde{x}_n \mid \boldsymbol{\rho}_k)]$ to derive the variational E and M steps.

The log probability density for the categorical variable with J levels indexed by $j = 1, \dots, J$ can be written

$$\log p(\tilde{x}_n \mid \boldsymbol{\rho}_k) = \log \prod_{j=1}^J \rho_k^{[[\tilde{x}_n=j]]} \quad (8.10)$$

$$= \sum_{j=1}^J \log \rho_{k,j}^{[[\tilde{x}_n=j]]} \quad (8.11)$$

$$= \sum_{j=1}^J [[\tilde{x}_n = j]] \log \rho_k, \quad (8.12)$$

where $[[\tilde{x}_n = j]] = 1$ when $\tilde{x}_n = j$ and zero elsewhere, and $\rho_{k,j}$ is the j -th element of $\boldsymbol{\rho}_k$.

The expected value is thus

$$\mathbb{E} \left[\sum_{j=1}^J [[\tilde{x}_n = j]] \log \rho_{k,j} \right] = \sum_{j=1}^J [[\tilde{x}_n = j]] \mathbb{E}[\log \rho_{k,j}] \quad (8.13)$$

with the standard result [19]

$$\mathbb{E}[\log \rho_{k,j}] = \psi(\gamma_{k,j}) - \psi(\hat{\gamma}_k) \quad (8.14)$$

where $\psi(\cdot)$ is the digamma function, $\hat{\gamma}_k = \sum_{j=1}^J \gamma_{k,j}$, and the $\gamma_{k,j}$ are computed in the variational M-step.

The variational M-step for the normally-distributed components is given in Chapter 2.

Conjugacy of the categorical and Dirichlet distributions enables the following parameter update [19]:

$$\gamma_{k,j} = \gamma_0 + \sum_{n=1}^N q(z_n = k)[[\tilde{x}_n = j]] \quad (8.15)$$

for $k = 1, \dots, K$ and $j = 1, \dots, J$. These variational E and M steps for the categorical variables can be computed alongside the variational E and M steps for the Gaussian variables in Algorithm 3.1 for online clustering with a BNP prior. However, additional work is necessary to use the automatic hyperparameter tuning methods discussed in Chapter 4.

8.2 Computing NISE for mixed data models

In order to employ the hyperparameter tuning methods proposed in Chapter 4, we must be able to compute the NISE between mixture models with mixed data types. We derive the un-normalized ISE here, which can be inserted into (4.14) to compute the NISE. For two collapsed mixture models p and q parameterized respectively by Θ and Θ'

$$p(\mathbf{x} \mid \Theta) = \frac{1}{K} \sum_{k=1}^K \text{Norm}(\check{\mathbf{x}}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \prod_{c=1}^C \text{Cat}(\tilde{x}_c; \boldsymbol{\rho}_{k,c}) \quad (8.16)$$

$$q(\mathbf{x} \mid \Theta') = \frac{1}{K'} \sum_{k=1}^{K'} \text{Norm}(\check{\mathbf{x}}; \boldsymbol{\mu}'_k, \boldsymbol{\Sigma}'_k) \prod_{c=1}^C \text{Cat}(\tilde{x}_c; \boldsymbol{\rho}'_{k,c}), \quad (8.17)$$

recall that the integrated squared error is defined as

$$\text{ISE}(\Theta, \Theta') = \int_{\mathcal{X}} (p(\mathbf{x} \mid \Theta) - q(\mathbf{x} \mid \Theta'))^2 d\mathbf{x} \quad (8.18)$$

$$= \int_{\mathcal{X}} p(\mathbf{x} \mid \Theta)^2 - 2p(\mathbf{x} \mid \Theta)q(\mathbf{x} \mid \Theta') + q(\mathbf{x} \mid \Theta')^2 d\mathbf{x} \quad (8.19)$$

$$= J_{pp} - 2J_{pq} + J_{qq}, \quad (8.20)$$

where

$$J_{pp} = \int_{\mathcal{X}} p(\mathbf{x} \mid \Theta)^2 d\mathbf{x} \quad (8.21)$$

$$J_{pq} = \int_{\mathcal{X}} p(\mathbf{x} \mid \Theta) q(\mathbf{x} \mid \Theta') d\mathbf{x} \quad (8.22)$$

$$J_{qq} = \int_{\mathcal{X}} q(\mathbf{x} \mid \Theta')^2 d\mathbf{x} \quad (8.23)$$

Note that we are using the covariance matrix Σ instead of the precision matrix $\mathbf{T}$.

In the following, we omit the conditional on Θ and Θ' to (slightly) simplify the notation. Working with (8.22) and noting that the independence of the Gaussian and categorical elements means they are separable, we have

$$\begin{aligned} \int_{\mathcal{X}} p(\mathbf{x}) q(\mathbf{x}) d\mathbf{x} &= \int_{\check{\mathcal{X}}} \left[\frac{1}{K} \sum_{k=1}^K \text{Norm}(\check{\mathbf{x}}; \boldsymbol{\mu}_k, \Sigma_k) \sum_{\tilde{\mathbf{x}}} \prod_{c=1}^C \text{Cat}(\tilde{x}_c; \boldsymbol{\rho}_{k,c}) \right. \\ &\quad \times \left. \frac{1}{K'} \sum_{\ell=1}^{K'} \text{Norm}(\check{\mathbf{x}}; \boldsymbol{\mu}'_{\ell}, \Sigma'_{\ell}) \sum_{\tilde{\mathbf{x}}} \prod_{c=1}^C \text{Cat}(\tilde{x}_c; \boldsymbol{\rho}'_{\ell,c}) \right] d\check{\mathbf{x}} \end{aligned} \quad (8.24)$$

$$\begin{aligned} &= \frac{1}{KK'} \sum_{k=1}^K \sum_{\ell=1}^{K'} \left[\int_{\check{\mathcal{X}}} \text{Norm}(\check{\mathbf{x}}; \boldsymbol{\mu}_k, \Sigma_k) \text{Norm}(\check{\mathbf{x}}; \boldsymbol{\mu}'_{\ell}, \Sigma'_{\ell}) d\check{\mathbf{x}} \right. \\ &\quad \times \left. \prod_{c=1}^C \sum_{\tilde{x}_c} \text{Cat}(\tilde{x}_c; \boldsymbol{\rho}_{k,c}) \text{Cat}(\tilde{x}_c; \boldsymbol{\rho}'_{\ell,c}) \right]. \end{aligned} \quad (8.25)$$

From [94] we know that

$$\int_{\check{\mathcal{X}}} \text{Norm}(\check{\mathbf{x}}; \boldsymbol{\mu}, \Sigma) \text{Norm}(\check{\mathbf{x}}; \boldsymbol{\mu}', \Sigma') d\check{\mathbf{x}} = \text{Norm}(\boldsymbol{\mu}; \boldsymbol{\mu}', \Sigma + \Sigma'). \quad (8.26)$$

For the categorical terms, note that

$$\text{Cat}(\tilde{x}; \boldsymbol{\rho}) \text{Cat}(\tilde{x}; \boldsymbol{\rho}') = \left[\prod_{j=1}^J \rho_j^{[[\tilde{x}=j]]} \right] \times \left[\prod_{j=1}^J (\rho'_j)^{[[\tilde{x}=j]]} \right] \quad (8.27)$$

$$= \prod_{j=1}^J (\rho_j \rho'_j)^{[[\tilde{x}=j]]}. \quad (8.28)$$

Since the support of a categorical distribution with J levels is $\mathcal{X} = \{1, \dots, J\}$, the summation of the product of categoricals in (8.25) can be stated by substituting (8.28):

$$\sum_{\tilde{x}} \text{Cat}(\tilde{x}; \boldsymbol{\rho}) \text{Cat}(\tilde{x}; \boldsymbol{\rho}') = \sum_{\tilde{x}=1}^J \prod_{j=1}^J (\rho_j \rho'_j)^{[[\tilde{x}=j]]} \quad (8.29)$$

$$= \sum_{j=1}^J \rho_j \rho'_j \quad (8.30)$$

$$= \boldsymbol{\rho}^\top \boldsymbol{\rho}'. \quad (8.31)$$

Plugging (8.26) and (8.31) into (8.25) yields

$$\int_{\mathcal{X}} p(\mathbf{x}) q(\mathbf{x}) d\mathbf{x} = \frac{1}{K K'} \sum_{k=1}^K \sum_{\ell=1}^{K'} \left[\text{Norm}(\boldsymbol{\mu}_k; \boldsymbol{\mu}'_\ell, \boldsymbol{\Sigma}_k + \boldsymbol{\Sigma}'_\ell) \prod_{c=1}^C \boldsymbol{\rho}_{k,c}^\top \boldsymbol{\rho}'_{\ell,c} \right]. \quad (8.32)$$

Similar steps may be followed to derive (8.21) and (8.23). Thus the ISE for mixed Gaussian and categorical mixture models is given by (8.20) where

$$J_{pp} = \frac{1}{K^2} \sum_{k=1}^K \sum_{\ell=1}^K \left[\text{Norm}(\boldsymbol{\mu}_k; \boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_k + \boldsymbol{\Sigma}_\ell) \prod_{c=1}^C \boldsymbol{\rho}_{k,c}^\top \boldsymbol{\rho}_{\ell,c} \right] \quad (8.33)$$

$$J_{pq} = \frac{1}{K K'} \sum_{k=1}^K \sum_{\ell=1}^{K'} \left[\text{Norm}(\boldsymbol{\mu}_k; \boldsymbol{\mu}'_\ell, \boldsymbol{\Sigma}_k + \boldsymbol{\Sigma}'_\ell) \prod_{c=1}^C \boldsymbol{\rho}_{k,c}^\top \boldsymbol{\rho}'_{\ell,c} \right] \quad (8.34)$$

$$J_{qq} = \frac{1}{(K')^2} \sum_{k=1}^{K'} \sum_{\ell=1}^{K'} \left[\text{Norm}(\boldsymbol{\mu}'_k; \boldsymbol{\mu}'_\ell, \boldsymbol{\Sigma}'_k + \boldsymbol{\Sigma}'_\ell) \prod_{c=1}^C (\boldsymbol{\rho}'_{k,c})^\top \boldsymbol{\rho}'_{\ell,c} \right]. \quad (8.35)$$

The NISE can then be computed from (4.14). This closed-form solution for the NISE between two mixed data mixture models enables the use of Algorithm 4.1 for mixed data, which was already demonstrated for purely Gaussian data in Chapters 5 to 7.

8.3 Experiments with mixed-type data

We now demonstrate the use of Algorithm 4.1 for mixed data, leveraging the results in the previous two sections of this chapter. For these experiments, we adopt a qualitative data mining or data exploration application. This is distinct from the other evaluations performed in this work, which thus far have focused on comparing a clustering result to ground truth, allowing quantitative comparisons between algorithm configurations. However, in practice, clustering algorithms are often employed to extract interesting knowledge and trends from data with decreased emphasis on accuracy measures with respect to ground truth (such as class labels) [41], [142]. In this way, clustering algorithm is a “lens” through which a user can examine the data. The algorithm extracts information that is consistent with its model, so the type of model employed for data mining applications can shed new light on a dataset and reveal trends that other models could not.

We also note that in the following analyses, the true number of clusters in the data is not known because the datasets are taken from real-world data. The numerical features are not likely to be Gaussian distributed, especially when conditioned on the class label (if it is even available). Therefore the primary interest is in the cluster *parameters* rather than the cluster *assignments* of individual data points. Over-clustering may still be a problem, however—it is desirable to recover a model with little redundancy—so hyperparameter tuning is still relevant.

Presenting results from mixed-data clustering can be challenging without knowing how to interpret them. Categorical parameters are encouraged by a strong prior to concentrate in a single category. However, this does not always occur. A cluster may be split

between two or more categories. For instance, consider four different sets of categorical parameters with the same number of levels: $\rho_1 = [1, 0, 0, 0, 0]$, $\rho_2 = [0.9, 0.1, 0.0, 0.0, 0.0]$, $\rho_3 = [0.6, 0.2, 0.05, 0.05, 0.1]$, and $\rho_4 = [0.2, 0.2, 0.2, 0.2, 0.2]$. Although the first level obviously dominates in the first three parameters, the difference between them is hard to quantify. On the other hand, ρ_4 is clearly spread evenly between all of the categories—a much less decisive result than the first three.

A quantitative representation of the concentration of ρ into a single category will assist in interpreting categorical parameters. To this end, we compute a concentration score

$$h(\rho) = 1 - \frac{H(\rho)}{H(\text{Unif}(J))}, \quad (8.36)$$

where H is the typical Shannon entropy and $\text{Unif}(J)$ is the discrete uniform distribution over 0 to $J - 1$. The ratio $H(\rho)/H(\text{Unif}(J)) = 1$ when categories in ρ are weighted uniformly and decreases as one or more categories become dominant. Subtracting this ratio from unity as in (8.36) yields a score between 0 and 1 where highly-concentrated categorical variables ρ have a higher score.

The concentration scores for the categorical distributions in the previous example (see Table 8.1) are $h(\rho_1) = 1$, $h(\rho_2) \approx 0.71$, $h(\rho_3) \approx 0.28$, and $h(\rho_4) = 0$, indicating that the categorical parameter is concentrated purely in one category for ρ_1 , multiple categories for ρ_2 and ρ_3 , and evenly distributed between all categories for ρ_4 . Note that $h(\rho_2) > h(\rho_3)$, reflecting that ρ_2 is more concentrated on a single category. Also, the concentration score is invariant to permutation of the categories, e.g., $h([0.9, 0.1]) = h([0.1, 0.9]) \approx 0.53$. When presenting results, we report the concentration score only if $h(\rho) < 1$. In this case, the top one (when $J = 2$) or two (when $J > 2$) categories are reported along with the concentration score. When no concentration score is reported, it is assumed that $h(\rho) = 1$, i.e., that the weights of the categorical variable are concentrated entirely in one level. As an example, if the categories in the example are labeled A through E , the result for all ρ_i

Table 8.1: Presentation of example results for categorical variables. When analyzing real datasets, only the Top 2 (h) column is reported. When $h(\rho) = 1$, its value is not given in parentheses.

Name	Parameter values	Top 2 (h)
ρ_1	[1, 0, 0, 0, 0]	A
ρ_2	[0.9, 0.1, 0.0, 0.0, 0.0]	A, B (0.71)
ρ_3	[0.6, 0.2, 0.05, 0.05, 0.1]	A, B (0.28)
ρ_4	[0.2, 0.2, 0.2, 0.2, 0.2]	A, B (0.00)

Table 8.2: Features extracted from the heart disease dataset.

Feature	Variable type	Categories
Resting blood pressure	Numerical	—
Maximum heart rate	Numerical	—
Serum cholesterol	Numerical	—
Fasting blood glucose	Categorical	$> 120, < 120$
Resting EKG result	Categorical	A, B, C

for $i = 1, \dots, 4$ is presented in the rightmost column of Table 8.1.

8.3.1 Heart disease

We apply the proposed BNP mixed-data clustering algorithm to analyze a dataset taken from the UCI machine learning database [143] describing 303 heart disease patients. Following [144], we work on a subset of features from the processed Cleveland dataset. Table 8.2 provides the list of features analyzed for this work, which includes three continuous variables and two categorical variables. Fig. 8.1 plots each of the numerical variables against one another. By inspection, it appears that the numerical features essentially form one large cluster. Incorporating categorical features may induce further separability, motivating the use of a mixed data model. Fig. 8.2 contains histograms for each of the categorical variables. The majority of patients in the dataset have a fasting blood sugar greater than

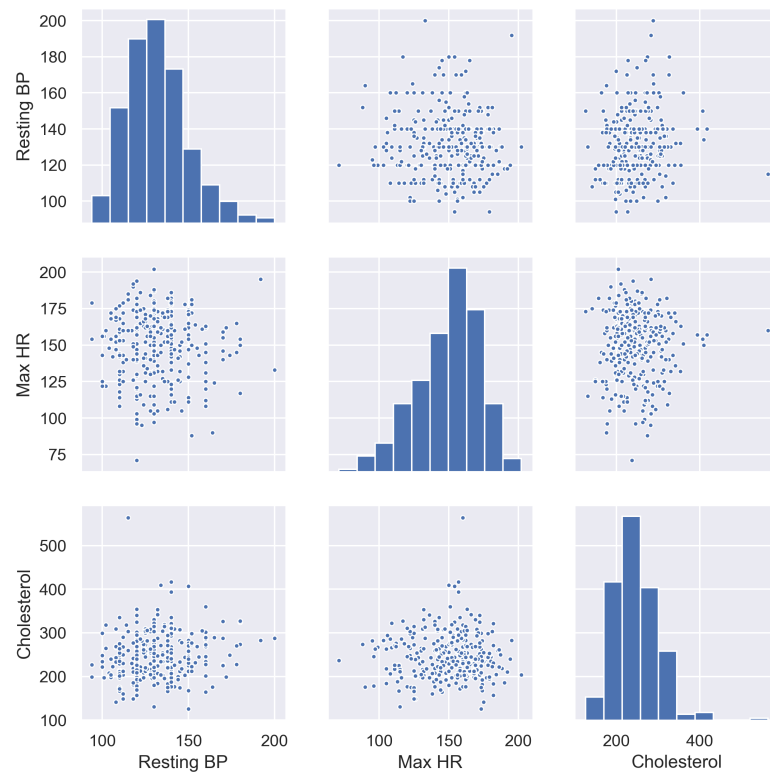


Figure 8.1: Numerical variables in the heart disease dataset plotted pairwise, with feature histograms plotted on the diagonals.

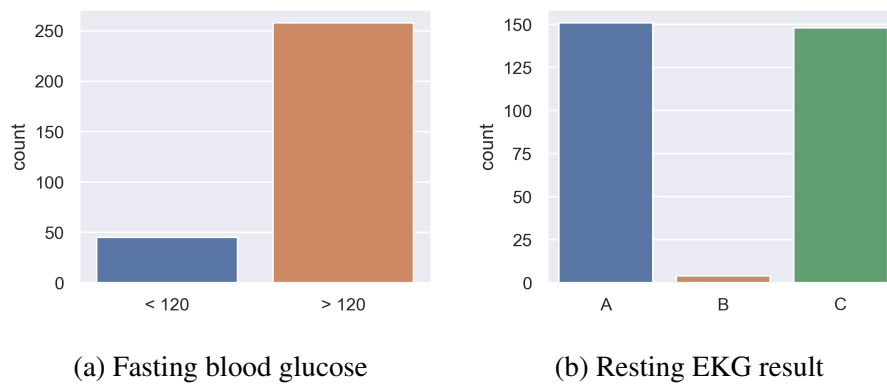


Figure 8.2: Histograms for the categorical features in the heart disease dataset.

Table 8.3: Prior hyperparameters for the mixed-data model used in heart disease clustering. A $D \times D$ identity matrix is denoted in this table by $\mathbf{I}$, where D is the number of numerical features. A vector of D zeros is denoted by $\mathbf{0}$. A vector of J ones is denoted by $\mathbf{1}_J$, where J is the length of the vector.

Hyperparameter	Value
$\mathbf{m}_0$	$\mathbf{0}$
ν_0	0.1
$\mathbf{W}_0$	$0.01\mathbf{I}$
λ_0	$\mathbf{D}$
$\gamma_{0,\text{BG}}$	$\mathbf{1}_2 \times 10^{-5}$
$\gamma_{0,\text{EKG}}$	$\mathbf{1}_3 \times 10^{-5}$

120, and most patients have a resting EKG result of A or C .

We split the data into two minibatches and cluster the data with our proposed mixed-data clustering algorithm configured for online processing with and without the root-finding tuning method ($\eta = 0.025$). We also cluster the data with a standard fully Gaussian mixture model (tuned via root-finding), with categorical features encoded directly from level to numeric value; e.g., the “resting EKG result” feature has three levels which are encoded numerically as ($A = 1, B = 2, C = 3$). *After clustering*, the numerical features are grouped into three equally sized bins and labeled High, Medium, and Low. Discretizing the results in this way eases interpretation of extracted clusters. The actual numerical values were used for clustering. Table 8.3 lists the hyperparameters used for the mixed-data model and fully Gaussian models. Prior hyperparameters for the fully Gaussian model also follow Table 8.3. We perform inference on all entries of the precision matrix (i.e., we do not constrain it to be diagonal) in order to capture potential correlations between numerical variables.

Table 8.4 lists the clusters found in the data with a mixed-data model and root-finding tuning method. Table 8.5 lists the clusters found in the data with a mixed-data model and no tuning ($r = 1.0$ fixed). Table 8.6 lists the clusters found in the data with a tuned fully Gaussian model. The fully Gaussian cluster centers for the categorical variables are

mapped to the nearest integer value.

The fully Gaussian model extracts a single cluster. Interestingly, the sole cluster has Resting EKG result B , despite this being the least common category in the data (see Fig. 8.2b). This is likely because of the way the categorical variable is encoded with $B = 2$, between $A = 1$ and $C = 3$ —a shortcoming of numerically encoding categorical variables instead of using a more intuitive mixed model (this problem is further demonstrated in Section 8.3.2). Comparing the tuned and untuned mixed-data results in Tables 8.4 and 8.5 reveals that tuning enables our algorithm to extract a more compact model with 5 clusters versus 9 in the untuned results. Note however that this comes at the cost of merging some data points with Fasting BG < 120 into clusters with Fasting BG > 120 (note the cluster in the fourth row of Table 8.4 has a concentration score $h(\rho) < 1$). On the other hand, the tuned mixed-data model extracts clusters that are more concentrated in the Resting EKG category compared to the untuned model.

Overall, the mixed data model extracts more information from the data and is easier to interpret. However, it is not immediately clear if tuning offers a significant advantage over using an untuned model. If model compactness is a priority, then tuning is likely to be a good choice. But tuning comes at the cost of potentially merging clusters that may provide valuable information.

8.3.2 Wine ratings

We use the proposed online BNP mixed-data clustering algorithm to analyze a dataset taken from Kaggle¹ containing wine reviews. The full dataset has around 150,000 entries that include two categorical features and two numerical features. This analysis covers a subset of around 35,000 entries with a limited selection of features. Table 8.8 provides the

¹Dataset available at <https://www.kaggle.com/zynicide/wine-reviews> as of 7 March 2020.

Table 8.4: Clusters found in the heart disease dataset with a mixed-data model and root-finding tuning method. The first three features are numerical, but were discretized into three equally sized bins *after* clustering in order to aid analysis. The last two features are categorical. For the “Fasting BG” feature, only the top category is reported, along with the concentration score (8.36) if it is less than one. The top two categories for “Resting EKG result” are reported along with the concentration score if it is less than one; otherwise, only the top category is reported.

Resting BP	Max HR	Cholesterol	Fasting BG ($J = 2$)	Resting EKG ($J = 3$)
Medium	Low	Low	> 120	B
High	Low	High	> 120	B
Low	High	Low	> 120	A
Medium	Medium	Medium	> 120 (0.37)	A/C (0.37)
Medium	Medium	High	> 120	C

Table 8.5: Clusters found in the heart disease dataset with a mixed-data model and no tuning. The first three features are numerical, but were discretized into three equally sized bins *after* clustering in order to aid analysis. The last two features are categorical. For the “Fasting BG” feature, only the top category is reported, along with the concentration score (8.36) if it is less than one. The top two categories for “Resting EKG result” are reported along with the concentration score if it is less than one; otherwise, only the top category is reported.

Resting BP	Max HR	Cholesterol	Fasting BG ($J = 2$)	Resting EKG ($J = 3$)
Low	Medium	Low	> 120	A
Medium	High	High	> 120 (0.39)	C
Low	Low	High	> 120	C/A (0.39)
Medium	High	Medium	> 120	A/C (0.60)
Medium	Medium	Low	< 120	A/C (0.59)
High	Low	Low	> 120	A/C (0.15)
Medium	Medium	High	> 120	C
Medium	Low	High	> 120 (0.32)	A/C (0.31)
High	Medium	Medium	> 120 (0.04)	C

Table 8.6: Clusters found in the heart disease dataset with a fully Gaussian model and root-finding tuning. The first three features are numerical, but were discretized into three equally sized bins *after* clustering in order to aid analysis. The last two features are categorical, but were encoded as numerical in order to fit the Gaussian model. The cluster centers for the categorical features were assigned to the category corresponding to the nearest integer.

Resting BP	Max HR	Cholesterol	Fasting BG ($J = 2$)	Resting EKG ($J = 3$)
Medium	Medium	Medium	> 120	B

Table 8.7: Encoding schemes used for categorical variables “Country” and “Variety” in the wine ratings dataset.

Category name	Encoding	
	A	B
Country		
France	1	2
Spain	2	3
USA	3	1
Variety		
Cabernet Sauvignon	1	2
Chardonnay	2	3
Pinot Noir	3	4
Red Blend	4	5
Riesling	5	6
Sauvignon Blanc	6	1

list of features analyzed for this work, which includes two continuous variables and two categorical variables. Fig. 8.3 plots each of the numerical variables against one another. Visual inspection of the data reveals that while the rating feature is somewhat Gaussian in shape (although it is an integer), the price is not. Fig. 8.4 contains histograms for each of the categorical variables. The majority of wines in the dataset are from the US, while the most frequent varieties are Cabernet Sauvignon, Chardonnay, and Pinot Noir.

In the previous section, we used the heart disease data to demonstrate how a mixed model can extract more useful information from a data with mixed types and to examine the impact of hyperparameter tuning. With the wine ratings dataset, we demonstrate that a mixed model is invariant to permutation of the levels in categorical features, while a standard fully Gaussian model is not. To do this, we cluster the dataset under the two categorical encoding schemes given in Table 8.7.

We split the data into 50 minibatches and cluster it with our proposed mixed-data clustering algorithm configured for online processing with the root-finding tuning method

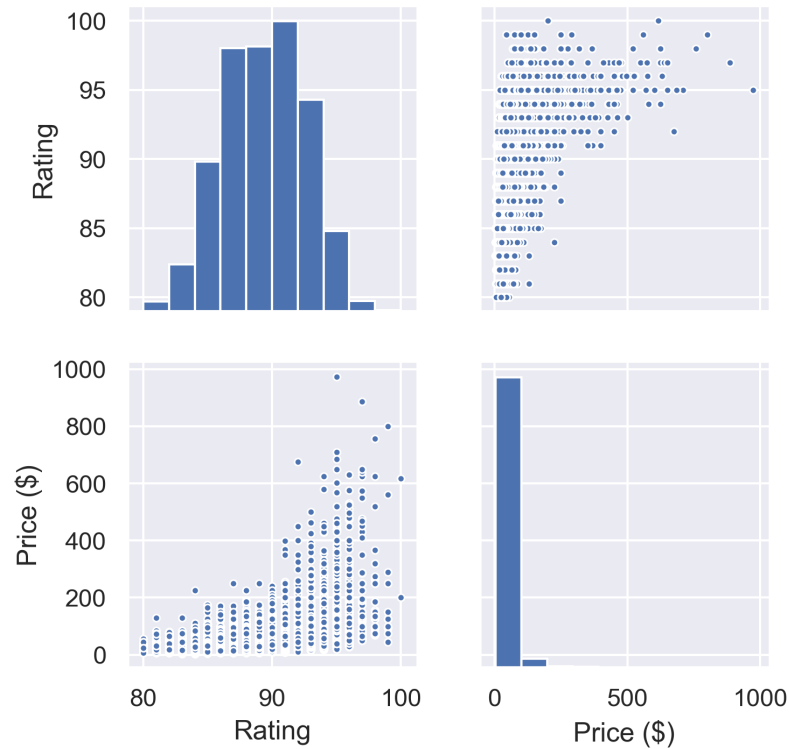


Figure 8.3: Numerical variables in the wine ratings dataset plotted pairwise, with feature histograms plotted on the diagonals.

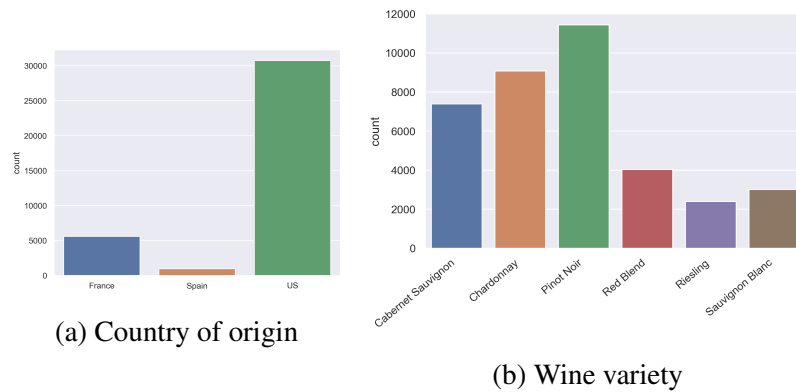


Figure 8.4: Histograms for the categorical features in the wine ratings dataset.

Table 8.8: Features extracted from the wine ratings dataset.

Feature	Variable type	Categories
Price	Numerical	—
Score	Numerical	—
Country	Categorical	France, Spain, US
Variety	Categorical	Cabernet Sauvignon, Chardonnay, Pinot Noir, Red Blend, Riesling, Sauvignon Blanc

Table 8.9: Prior hyperparameters for the mixed-data model used in wine ratings clustering. A $D \times D$ identity matrix is denoted in this table by $\mathbf{I}$, where D is the number of numerical features. A vector of D zeros is denoted by $\mathbf{0}$. A vector of J ones is denoted by $\mathbf{1}_J$, where J is the length of the vector.

Hyperparameter	Value
$\mathbf{m}_0$	$\mathbf{0}$
ν_0	0.1
$\mathbf{W}_0$	$\mathbf{I}$
λ_0	D
$\gamma_{0,\text{Country}}$	$\mathbf{1}_3 \times 10^{-5}$
$\gamma_{0,\text{Variety}}$	$\mathbf{1}_6 \times 10^{-5}$

($\eta = 0.025$). We also cluster the data with a standard fully Gaussian mixture model (tuned via root-finding), with categorical features encoded as given in Table 8.7. Table 8.9 lists the hyperparameters used for the mixed-data model. Prior hyperparameters for the fully Gaussian model also follow Table 8.9. Like with the heart disease data, we perform inference on all entries of the precision matrix.

Table 8.10 lists the clusters found in the data with a mixed-data model and root-finding tuning method. Table 8.11 lists the clusters found in the data with a tuned fully Gaussian model for each of the encodings. The fully Gaussian cluster centers for the categorical variables are mapped to the nearest integer value.

The different results from the fully Gaussian model in Table 8.11 highlight a shortcoming of naive numerical encoding of categorical features. When categorical features are modeled numerically, permuting the encoding changes the result significantly. This is

Table 8.10: Clusters found in the wine ratings dataset with a mixed-data model and root-finding tuning method. The first two features are numerical. The last two features are categorical. The top two categories for the categorical features are reported along with the concentration score (8.36) if it is less than one; otherwise, only the top category is reported. Results were identical for each of the encodings in Table 8.7 due to the explicitly unordered modeling of categorical variables “country” and “variety”.

Rating	Price (\$)	Variety ($J = 6$)	Country ($J = 3$)
89.0	31.65	Pinot Noir/Cabernet Sauvignon (0.412)	France/Spain (0.470)
90.1	45.02	Red Blend	France
90.1	59.56	Chardonnay	France
88.5	35.41	Riesling	France/US (0.551)
89.9	73.74	Red Blend	Spain

undesirable because categorical features are unordered—artificially imposing an order on these features can couple the solution to the encoding, leading to unstable results. On the other hand, properly modeling categories as unordered symbols, as we have done here, yields a stable result as given in Table 8.10.

8.4 Summary

Real-world datasets often have features with mixed numerical and discrete categorical types. Typically, the data is encoded in a way that transforms all of the categorical features into numerical ones in order to promote compatibility with standard clustering algorithms. However, this is not always effective. For example, if unordered categories are encoded as ordered integers, the clustering solution may become unstable depending on the mapping.

In this chapter, we showed that the model-based approach proposed in this dissertation can be extended to cluster data with mixed type in a more principled manner. We demonstrated that our algorithm can extract an expressive, yet succinct description of a heart disease dataset with less redundancy than a more typical, CRP based BNP clustering algorithm. We also showed that treating categorical features properly (as opposed to

Table 8.11: Clusters found in the wine ratings dataset with a fully Gaussian model and root-finding tuning. Encodings A and B are given in Table 8.7. The first two features are numerical. The last two features are categorical, but were encoded as numerical in order to fit the Gaussian model. The cluster centers for the categorical features were assigned to the category corresponding to the nearest integer. Note that depending on the encoding, the cluster results are different (including the estimated number of clusters).

Rating	Price (\$)	Variety ($J = 6$)	Country ($J = 3$)
Encoding A			
88.1	22.42	Red Blend	France
89.3	33.26	Pinot Noir	France
90.2	61.71	Chardonnay	France
89.6	36.18	Riesling	US
90.1	45.23	Red Blend	France
90.7	95.86	Riesling	France
Encoding B			
89.1	33.61	Chardonnay	Spain
91.5	128.14	Pinot Noir	Spain
89.6	36.18	Riesling	US
89.5	38.21	Red Blend	France
90.1	60.20	Chardonnay	France

numerically encoding them) produces stable clustering results on a large dataset of wine reviews. These qualitative results indicate that a model-based clustering algorithm like the one proposed in this dissertation is a more appropriate choice for clustering mixed data than arbitrarily encoding categorical features.

Chapter 9

Conclusion

In this dissertation, we derived a new online Bayesian nonparametric clustering method which provides better estimates of the number of clusters in the data than the state of the art. We showed that the proposed approach can adapt to dynamically changing datasets, making it suitable for online clustering tasks such as radar pulse clustering. We also demonstrated that the model-based framework is extensible for non-Gaussian data types. In this chapter, we summarize the contributions of this work, comment on its expected impact, and provide suggestions for future work.

9.1 Contributions

This dissertation comprises four contributions:

1. **A novel online BNP clustering algorithm specifically designed to provide better model order estimates than the state of the art (Chapter 3).** We leverage the recently developed Powered Chinese Restaurant Process prior which helps control redundancy in BNP clustering in combination with the variational inference framework. Fast online inference is enabled by a second-order Taylor approximation for

the expected log-PCR_P.

2. **Two new hyperparameter tuning methods that enable the clustering algorithm to adapt to changes in the data stream (Chapter 4).** The PCR_P relies on a critical hyperparameter in order to control the tendency to over or under estimate the number of clusters. We combine techniques from Gaussian mixture reduction and semi-parametric change detection to tune the hyperparameter online without supervision.
3. **Empirical analyses of the algorithm and tuning methods for a variety of datasets (Chapters 5 to 7).** We show that our collapsed Gaussian mixture model can estimate the number of clusters in dynamically changing datasets better than the standard BNP approach. Three radar pulse clustering simulations indicate that our algorithm also outperforms the state of the art online BNP clustering algorithm in this application area. Finally, our approach also provides competitive results on neural spike sorting data when compared to other clustering algorithms specifically designed for that task—with the addition of efficient online processing capability.
4. **An extension of the algorithm to cluster data with mixed Gaussian and categorical types (Chapter 8).** We derived a mixed Gaussian-categorical model for online BNP clustering with our algorithm framework. We also extended the NISE measure to the mixed-data case in order to leverage our hyperparameter tuning methods. The proposed mixed data mixture model is more stable than a GMM for clustering data of mixed type. The resulting algorithm was employed for a data mining task with two real-world mixed-type datasets.

9.2 Expected impact

BNP clustering algorithms offer a way to sidestep the requirement that the number of clusters in the data is known a priori and fixed while processing data. However, it has been

shown (both here and elsewhere [27], [29]) that the inferred number of clusters is not always correct. Recent work has addressed this issue for static datasets, but prior to the work presented here, no viable solutions had been proposed for dynamic online clustering applications, where cluster statistics and the number of clusters can change over time. Online clustering applications like radar pulse processing—as demonstrated in this dissertation—require methods that efficiently process constant streams of data without redundancy. We expect that our work will be useful for these kinds of applications. Additionally, we believe that the general problem of online hyperparameter tuning is not unique to our algorithm. Our contributions to unsupervised hyperparameter selection may be useful for other clustering approaches that utilize probabilistic models.

9.3 Future work

We identify the following avenues for follow-on work:

- The performance of a model-based clustering algorithm may degrade with the introduction of outlier points that do not fit the proposed generative model. Methods for handling outliers exist (e.g., [145]–[147]), but have yet to be explored within our framework, particularly for dynamically changing datasets. Future efforts should characterize the sensitivity of the algorithm to outliers and explore ways to make the proposed clustering algorithm more robust.
- Our work in Chapter 4 revealed that online hyperparameter tuning without a supervisory signal is a difficult problem. The methods we propose were shown to be effective in Chapters 5 to 8, but we note that there is significant computational cost incurred by the proposed methods. It is important that the cost of tuning the hyperparameter does not forfeit the computational advantages we gain from online VI. Future work should focus on improving the speed of the tuning algorithms or proposing more efficient methods.

- An advantage of our optimization-based tuning method is that it does not rely on any user-selected hyper-hyperparameters. However, its derivation is heavily based on empirical analysis of the distribution of the number of clusters given the hyperparameter $p(K \mid r)$. It relies on a parameterization of a Poisson distribution that is itself an approximation. Future work on the PCR and its generalization, the GCRP could seek analytical forms for $p(K \mid r)$ which may improve the hyperparameter tuning process. Furthermore, our approximation largely ignores the influence of the number of data points N and the CRP concentration hyperparameter α . Possible next steps could incorporate these values for improved inference.
- We leveraged the NISE measure from Gaussian mixture reduction in order to find a reduced GMM that is “similar” to the over-clustered GMM, but with fewer clusters. However, our algorithm could also be used for more typical Gaussian mixture reduction applications, where the size of the reduced model is known a priori. The PCR hyperparameter could be used as a parameter to reduce the number of components in a GMM until the specified number of clusters is achieved. Further investigation is needed to determine the validity of this approach.
- In Chapters 7 and 8, we divided single datasets into multiple minibatches to illustrate the online processing paradigm and process the data more efficiently. After the final minibatch was processed, all of the minibatches of data were passed through the algorithm a final time (without modifying the cluster parameters) to assign each datapoint to a cluster. This means that the last minibatch may have greater influence on the final cluster parameters than prior minibatches, i.e., the cluster parameters may have changed significantly from the first minibatches. Future work could investigate better ways to utilize cluster parameters from all minibatches when making a final cluster assignment for each datapoint, e.g., incorporating methods from Bayesian filtering or smoothing [148].

References

- [1] A. K. Jain, R. C. Dubes, *et al.*, *Algorithms for clustering data*. Prentice hall Englewood Cliffs, 1988, vol. 6.
- [2] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344.
- [3] B. S. Everitt, S. Landau, M. Leese, and D. Stahl, *Cluster Analysis*. John Wiley & Sons, Ltd, Jan. 2011. DOI: [10 . 1002 / 9780470977811](https://doi.org/10.1002/9780470977811). [Online]. Available: <https://doi.org/10.1002/9780470977811>.
- [4] K. Lee, L. Guillemot, Y. Yue, M. Kramer, and D. Champion, “Application of the gaussian mixture model in pulsar astronomy-pulsar classification and candidates ranking for the fermi 2fgl catalogue,” *Monthly Notices of the Royal Astronomical Society*, vol. 424, no. 4, pp. 2832–2840, 2012.
- [5] K. Todros and J. Tabrikian, “Blind separation of independent sources using gaussian mixture model,” *IEEE transactions on signal processing*, vol. 55, no. 7, pp. 3645–3658, 2007.

- [6] Z. Guo, D. Shi, D. E. Quevedo, and L. Shi, "Secure state estimation against integrity attacks: A gaussian mixture model approach," *IEEE Transactions on Signal Processing*, vol. 67, no. 1, pp. 194–207, 2019.
- [7] T. Chappell, S. Geva, and J. Hogan, "K-means clustering of biological sequences," in *Proceedings of the 22Nd Australasian Document Computing Symposium*, ser. ADCS 2017, Brisbane, QLD, Australia: ACM, 2017, 2:1–2:4, ISBN: 978-1-4503-6391-4. DOI: [10.1145/3166072.3166076](https://doi.org/10.1145/3166072.3166076). [Online]. Available: <http://doi.acm.org/10.1145/3166072.3166076>.
- [8] K. Subramani, A. Velkov, I. Ntoutsis, P. Kroger, and H.-P. Kriegel, "Density-based community detection in social networks," in *2011 IEEE 5th International Conference on Internet Multimedia Systems Architecture and Application*, IEEE, 2011, pp. 1–8.
- [9] X. Duan, F. Yang, E. Antono, W. Yang, P. Pianetta, S. Ermon, A. Mehta, and Y. Liu, "Unsupervised data mining in nanoscale x-ray spectro-microscopic study of ndfeb magnet," *Scientific reports*, vol. 6, p. 34 406, 2016.
- [10] B. Thomas, G. Raju, and S. Wangmo, "A modified fuzzy c-means algorithm for natural data exploration," *World Academy of science, Engineering and technology*, vol. 49, pp. 478–481, 2009.
- [11] D. S. Johnson and E. H. Sinclair, "Modeling joint abundance of multiple species using dirichlet process mixtures," *Environmetrics*, vol. 28, no. 3, e2440, Mar. 2017. DOI: [10.1002/env.2440](https://doi.org/10.1002/env.2440). [Online]. Available: <https://doi.org/10.1002/env.2440>.

- [12] B. J. Reich and H. D. Bondell, "A spatial dirichlet process mixture model for clustering population genetics data," *Biometrics*, vol. 67, no. 2, pp. 381–390, Sep. 2010. DOI: [10.1111/j.1541-0420.2010.01484.x](https://doi.org/10.1111/j.1541-0420.2010.01484.x). [Online]. Available: <https://doi.org/10.1111/j.1541-0420.2010.01484.x>.
- [13] A. R. F. da Silva, "Bayesian mixture models of variable dimension for image segmentation," *Computer Methods and Programs in Biomedicine*, vol. 94, no. 1, pp. 1–14, Apr. 2009. DOI: [10.1016/j.cmpb.2008.05.010](https://doi.org/10.1016/j.cmpb.2008.05.010). [Online]. Available: <https://doi.org/10.1016/j.cmpb.2008.05.010>.
- [14] K. M. Hammouda and M. S. Kamel, "Efficient phrase-based document indexing for web document clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 10, pp. 1279–1296, Oct. 2004, ISSN: 1041-4347. DOI: [10.1109/TKDE.2004.58](https://doi.org/10.1109/TKDE.2004.58).
- [15] J. Mei, Y. Wang, L. Chen, and C. Miao, "Large scale document categorization with fuzzy clustering," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 5, pp. 1239–1251, Oct. 2017, ISSN: 1063-6706. DOI: [10.1109/TFUZZ.2016.2604009](https://doi.org/10.1109/TFUZZ.2016.2604009).
- [16] S.-H. Oh, J.-S. Kang, Y.-C. Byun, G.-L. Park, and S.-Y. Byun, "Intrusion detection based on clustering a data stream," in *Third ACIS Int'l Conference on Software Engineering Research, Management and Applications (SERA'05)*, IEEE, 2005, pp. 220–227.
- [17] J. Liu, J. P. Lee, L. Li, Z.-Q. Luo, and K. M. Wong, "Online clustering algorithms for radar emitter classification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 8, pp. 1185–1196, 2005.

- [18] D. Sculley, “Web-scale k-means clustering,” in *Proceedings of the 19th international conference on World wide web*, ACM, 2010, pp. 1177–1178.
- [19] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006, ISBN: 0387310738.
- [20] H. Attias, “A variational bayesian framework for graphical models,” in *Advances in neural information processing systems*, 2000, pp. 209–215.
- [21] Z. Zhang, K. L. Chan, Y. Wu, and C. Chen, “Learning a multivariate gaussian mixture model with the reversible jump mcmc algorithm,” *Statistics and Computing*, vol. 14, no. 4, pp. 343–355, 2004.
- [22] M. Aitkin and D. B. Rubin, “Estimation and hypothesis testing in finite mixture models,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 47, no. 1, pp. 67–75, 1985.
- [23] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the em algorithm,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.
- [24] H. Hruschka, “Market definition and segmentation using fuzzy clustering methods,” *International Journal of research in Marketing*, vol. 3, no. 2, pp. 117–134, 1986.
- [25] F. Valafar, “Pattern recognition techniques in microarray data analysis: A survey,” *ANNALS-NEW YORK ACADEMY OF SCIENCES*, vol. 980, pp. 41–64, 2002.

- [26] P. Orbanz and Y. W. Teh, “Baysian nonparametric models,” *Encyclopedia of machine learning*, no. 1, 2010.
- [27] J. W. Miller and M. T. Harrison, “A simple example of dirichlet process mixture inconsistency for the number of components,” in *Advances in neural information processing systems*, 2013, pp. 199–206.
- [28] A. Onogi, M. Nurimoto, and M. Morita, “Characterization of a bayesian genetic clustering algorithm based on a dirichlet process prior and comparison among bayesian clustering methods,” *BMC bioinformatics*, vol. 12, no. 1, p. 263, 2011.
- [29] J. Lu, M. Li, and D. Dunson, “Reducing over-clustering via the powered chinese restaurant process,” *arXiv preprint arXiv:1802.05392*, 2018.
- [30] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, “Variational inference: A review for statisticians,” *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.
- [31] V. Huynh and D. Phung, “Streaming clustering with bayesian nonparametric models,” *Neurocomputing*, vol. 258, pp. 52–62, 2017.
- [32] N. Ailon, R. Jaiswal, and C. Monteleoni, “Streaming k-means approximation,” in *Advances in neural information processing systems*, 2009, pp. 10–18.
- [33] M. R. Ackermann, M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler, “Streamkm++: A clustering algorithm for data streams,” *Journal of Experimental Algorithmics (JEA)*, vol. 17, pp. 2–4, 2012.

- [34] T. Broderick, N. Boyd, A. Wibisono, A. C. Wilson, and M. I. Jordan, “Streaming variational bayes,” in *Advances in Neural Information Processing Systems*, 2013, pp. 1727–1735.
- [35] D. M. Blei, M. I. Jordan, *et al.*, “Variational inference for dirichlet process mixtures,” *Bayesian analysis*, vol. 1, no. 1, pp. 121–143, 2006.
- [36] K. Kurihara, M. Welling, and Y. W. Teh, “Collapsed variational dirichlet process mixture models,” in *IJCAI*, vol. 7, 2007, pp. 2796–2801.
- [37] M. Moghaddasi, M. A. Shoorehdeli, Z. Fatahi, and A. Haghparast, “Unsupervised automatic online spike sorting using reward-based online clustering,” *Biomedical Signal Processing and Control*, vol. 56, p. 101 701, 2020.
- [38] M. Scherreik and B. Rigling, “Clustering radar pulses with Bayesian nonparametrics: A case for online processing,” in *2020 IEEE Radar Conference*, IEEE, 2020.
- [39] —, “Controlling redundancy in online Bayesian nonparametric clustering,” *IEEE Transactions on Signal Processing*, 2020, under review.
- [40] —, “Clustering radar pulses with online Bayesian nonparametrics for dynamic spectrum awareness,” *IEEE Transactions on Aerospace and Electronic Systems*, 2020, under review.
- [41] P.-N. Tan, *Introduction to data mining*. Pearson Education India, 2018.
- [42] Rui Xu and D. Wunsch, “Survey of clustering algorithms,” *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678, May 2005, ISSN: 1045-9227. DOI: [10.1109/TNN.2005.845141](https://doi.org/10.1109/TNN.2005.845141).

- [43] P. Berkhin, “A survey of clustering data mining techniques,” in *Grouping multidimensional data*, Springer, 2006, pp. 25–71.
- [44] D. Xu and Y. Tian, “A comprehensive survey of clustering algorithms,” *Annals of Data Science*, vol. 2, no. 2, pp. 165–193, Jun. 2015, ISSN: 2198-5812. DOI: [10.1007/s40745-015-0040-1](https://doi.org/10.1007/s40745-015-0040-1). [Online]. Available: <https://doi.org/10.1007/s40745-015-0040-1>.
- [45] J. MacQueen *et al.*, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Oakland, CA, USA, vol. 1, 1967, pp. 281–297.
- [46] L. Kauffman and P. Rousseeuw, “Clustering by means of medoids,” *Proc. Statistical Data Analysis Based on the L1 Norm Conference, Neuchatel, 1987*, pp. 405–416, 1987. [Online]. Available: <https://ci.nii.ac.jp/naid/10027761751/en/>.
- [47] T. Zhang, R. Ramakrishnan, and M. Livny, “Birch: An efficient data clustering method for very large databases,” in *ACM Sigmod Record*, ACM, vol. 25, 1996, pp. 103–114.
- [48] S. Guha, R. Rastogi, and K. Shim, “Cure: An efficient clustering algorithm for large databases,” in *ACM Sigmod Record*, ACM, vol. 27, 1998, pp. 73–84.
- [49] G. Karypis, E.-H. S. Han, and V. Kumar, “Chameleon: Hierarchical clustering using dynamic modeling,” *Computer*, no. 8, pp. 68–75, 1999.

- [50] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” AAAI Press, 1996, pp. 226–231.
- [51] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, “Optics: Ordering points to identify the clustering structure,” in *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD ’99, Philadelphia, Pennsylvania, USA: ACM, 1999, pp. 49–60, ISBN: 1-58113-084-8. DOI: [10.1145/304182.304187](https://doi.org/10.1145/304182.304187). [Online]. Available: <http://doi.acm.org/10.1145/304182.304187>.
- [52] K. Choromanski, S. Kumar, and X. Liu, “Fast online clustering with randomized skeleton sets,” *arXiv preprint arXiv:1506.03425*, 2015.
- [53] R. O. Duda, P. E. Hart, *et al.*, *Pattern classification and scene analysis*. Wiley New York, 1973, vol. 3.
- [54] R. M. Neal, “Bayesian mixture modeling,” in *Maximum Entropy and Bayesian Methods*, Springer, 1992, pp. 197–211.
- [55] C. E. Rasmussen, “The infinite gaussian mixture model,” in *Advances in neural information processing systems*, 2000, pp. 554–560.
- [56] R. Tibshirani, G. Walther, and T. Hastie, “Estimating the number of clusters in a data set via the gap statistic,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, May 2001. DOI: [10.1111/1467-9868.00293](https://doi.org/10.1111/1467-9868.00293). [Online]. Available: <https://doi.org/10.1111/1467-9868.00293>.

- [57] C. A. Sugar and G. M. James, “Finding the number of clusters in a dataset,” *Journal of the American Statistical Association*, vol. 98, no. 463, pp. 750–763, Sep. 2003. DOI: [10.1198/016214503000000666](https://doi.org/10.1198/016214503000000666). [Online]. Available: <https://doi.org/10.1198/016214503000000666>.
- [58] P. McCullagh, J. Yang, *et al.*, “How many clusters?” *Bayesian Analysis*, vol. 3, no. 1, pp. 101–120, 2008.
- [59] P. J. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, Nov. 1987. DOI: [10.1016/0377-0427\(87\)90125-7](https://doi.org/10.1016/0377-0427(87)90125-7). [Online]. Available: [https://doi.org/10.1016/0377-0427\(87\)90125-7](https://doi.org/10.1016/0377-0427(87)90125-7).
- [60] T. S. Ferguson, “A bayesian analysis of some nonparametric problems,” *The Annals of Statistics*, vol. 1, no. 2, pp. 209–230, Mar. 1973. DOI: [10.1214/aos/1176342360](https://doi.org/10.1214/aos/1176342360). [Online]. Available: <https://doi.org/10.1214/aos/1176342360>.
- [61] M. Shindler, A. Wong, and A. W. Meyerson, “Fast and accurate k-means for large datasets,” in *Advances in neural information processing systems*, 2011, pp. 2375–2383.
- [62] E. Liberty, R. Sriharsha, and M. Sviridenko, “An algorithm for online k-means clustering,” in *2016 Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, SIAM, 2016, pp. 81–89.
- [63] Y. Zhang, L. Chen, and X. Ran, “Online incremental em training of gmm and its application to speech processing applications,” in *IEEE 10th INTERNATIONAL CON-*

ERENCE ON SIGNAL PROCESSING PROCEEDINGS, IEEE, 2010, pp. 1309–1312.

- [64] A. Declercq and J. H. Piater, “Online learning of gaussian mixture models-a two-level approach,” in *VISAPP (I)*, 2008, pp. 605–611.
- [65] R. C. Pinto and P. M. Engel, “A fast incremental gaussian mixture model,” *PloS one*, vol. 10, no. 10, e0139931, 2015.
- [66] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan, “An introduction to MCMC for machine learning,” *Machine Learning*, vol. 50, no. 1/2, pp. 5–43, 2003. DOI: [10.1023/a:1020281327116](https://doi.org/10.1023/a:1020281327116). [Online]. Available: <https://doi.org/10.1023/a:1020281327116>.
- [67] R. Bardenet, A. Doucet, and C. Holmes, “On markov chain monte carlo methods for tall data,” *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 1515–1557, 2017.
- [68] C. P. Robert, V. Elvira, N. Tawn, and C. Wu, “Accelerating mcmc algorithms,” *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 10, no. 5, e1435, 2018.
- [69] P. R. Conrad, A. D. Davis, Y. M. Marzouk, N. S. Pillai, and A. Smith, “Parallel local approximation MCMC for expensive models,” *SIAM/ASA Journal on Uncertainty Quantification*, vol. 6, no. 1, pp. 339–373, Jan. 2018. DOI: [10.1137/16m1084080](https://doi.org/10.1137/16m1084080). [Online]. Available: <https://doi.org/10.1137/16m1084080>.

- [70] C. Wang, J. Paisley, and D. M. Blei, “Online variational inference for the hierarchical dirichlet process,” in *In Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS, AAAI, 2011*, pp. 752–760.
- [71] M. Bryant and E. B. Sudderth, “Truly nonparametric online variational inference for hierarchical dirichlet processes,” in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS’12, Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 2699–2707. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2999325.2999436>.
- [72] Y. W. Teh, “Dirichlet process,” *Encyclopedia of machine learning*, pp. 280–287, 2010.
- [73] M. West, *Hyperparameter estimation in Dirichlet process mixture models*. Duke University ISDS Discussion Paper# 92-A03, 1992.
- [74] R. M. Neal, “Markov chain sampling methods for dirichlet process mixture models,” *Journal of computational and graphical statistics*, vol. 9, no. 2, pp. 249–265, 2000.
- [75] M. J. Beal, “Variational algorithms for approximate bayesian inference,” University of London, Tech. Rep., 2003.
- [76] J. Bernardo, M. Bayarri, J. Berger, A. Dawid, D. Heckerman, A. Smith, M. West, *et al.*, “The variational bayesian em algorithm for incomplete data: With application to scoring graphical model structures,” *Bayesian statistics*, vol. 7, pp. 453–464, 2003.
- [77] D. J. Aldous, “Exchangeability and related topics,” in Springer, 1985, pp. 1–198.

- [78] Y. W. Teh, D. Newman, and M. Welling, “A collapsed variational bayesian inference algorithm for latent dirichlet allocation,” in *Advances in neural information processing systems*, 2007, pp. 1353–1360.
- [79] K. Wolter, *Introduction to variance estimation*. Springer Science & Business Media, 2007.
- [80] D. G. Tzikas, A. C. Likas, and N. P. Galatsanos, “The variational approximation for bayesian inference,” *IEEE Signal Processing Magazine*, vol. 25, no. 6, pp. 131–146, 2008.
- [81] A. Wu, S. Nowozin, E. Meeds, R. E. Turner, J. M. Hern’andez-Lobato, and A. L. Gaunt, “Deterministic variational inference for robust bayesian neural networks,” in *ICLR 2019*, 2019.
- [82] J. Altosaar, R. Ranganath, and D. Blei, “Proximity variational inference,” in *International Conference on Artificial Intelligence and Statistics*, 2018, pp. 1961–1969.
- [83] J. Blömer and K. Bujna, “Adaptive seeding for gaussian mixture models,” in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, 2016, pp. 296–308.
- [84] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, “API design for machine learning software: Experiences from the scikit-learn project,” in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

- [85] C. A. McGrory and D. Titterton, “Variational approximations in bayesian model selection for finite mixture distributions,” *Computational Statistics & Data Analysis*, vol. 51, no. 11, pp. 5352–5367, 2007.
- [86] A. Corduneanu and C. M. Bishop, “Variational bayesian model selection for mixture distributions,” in *Artificial intelligence and Statistics*, Morgan Kaufmann Waltham, MA, vol. 2001, 2001, pp. 27–34.
- [87] M. D. Escobar and M. West, “Bayesian density estimation and inference using mixtures,” *Journal of the american statistical association*, vol. 90, no. 430, pp. 577–588, 1995.
- [88] C. E. Antoniak, “Mixtures of dirichlet processes with applications to bayesian non-parametric problems,” *The annals of statistics*, pp. 1152–1174, 1974.
- [89] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, Í. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. 1. 0. Contributors, “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, 2020. DOI: <https://doi.org/10.1038/s41592-019-0686-2>.
- [90] W. K. Hastings, “Monte carlo sampling methods using markov chains and their applications,” 1970.

- [91] J. R. Hershey and P. A. Olsen, “Approximating the kullback leibler divergence between gaussian mixture models,” in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP’07*, IEEE, vol. 4, 2007, pp. IV–317.
- [92] Y. Bar-Yosef and Y. Bistriz, “Gaussian mixture models reduction by variational maximum mutual information,” *IEEE Transactions on Signal Processing*, vol. 63, no. 6, pp. 1557–1569, 2015.
- [93] A. Assa and K. N. Plataniotis, “Wasserstein-distance-based gaussian mixture reduction,” *IEEE Signal Processing Letters*, vol. 25, no. 10, pp. 1465–1469, 2018.
- [94] J. L. Williams, “Gaussian mixture reduction for tracking multiple maneuvering targets in clutter,” AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH, Tech. Rep., 2003.
- [95] D. J. Petrucci, “Gaussian mixture reduction for bayesian target tracking in clutter,” AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING AND . . . , Tech. Rep., 2005.
- [96] D. W. Scott and W. F. Szewczyk, “From kernels to mixtures,” *Technometrics*, vol. 43, no. 3, pp. 323–335, 2001.
- [97] D. F. Crouse, P. Willett, K. Pattipati, and L. Svensson, “A look at gaussian mixture reduction algorithms,” in *14th International Conference on Information Fusion*, IEEE, 2011, pp. 1–8.
- [98] J. Gama, *Knowledge discovery from data streams*. Chapman and Hall/CRC, 2010.

- [99] L. I. Kuncheva, “Change detection in streaming multivariate data using likelihood detectors,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 5, pp. 1175–1180, 2011.
- [100] N. X. Vinh, J. Epps, and J. Bailey, “Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance,” *Journal of Machine Learning Research*, vol. 11, no. Oct, pp. 2837–2854, 2010.
- [101] M. Song, C. Xin, Y. Zhao, and X. Cheng, “Dynamic spectrum access: From cognitive radio to network radio,” *IEEE Wireless Communications*, vol. 19, no. 1, pp. 23–29, Feb. 2012. DOI: [10.1109/mwc.2012.6155873](https://doi.org/10.1109/mwc.2012.6155873). [Online]. Available: <https://doi.org/10.1109/mwc.2012.6155873>.
- [102] C. de Souza Lima, F. Paisana, J. F. de Rezende, and L. A. DaSilva, “A cooperative approach for dynamic spectrum access in radar bands,” in *2014 International Telecommunications Symposium (ITS)*, IEEE, 2014, pp. 1–5.
- [103] F. Paisana, Z. Khan, J. Lehtomäki, L. A. DaSilva, and R. Vuoltoniemi, “Exploring radio environment map architectures for spectrum sharing in the radar bands,” in *2016 23rd International Conference on Telecommunications (ICT)*, IEEE, 2016, pp. 1–6.
- [104] H. T. Hayvaci and B. Tavli, “Spectrum sharing in radar and wireless communication systems: A review,” in *2014 International Conference on Electromagnetics in Advanced Applications (ICEAA)*, IEEE, Aug. 2014. DOI: [10.1109/iceaa.2014.6903969](https://doi.org/10.1109/iceaa.2014.6903969). [Online]. Available: <https://doi.org/10.1109/iceaa.2014.6903969>.

- [105] L. Zheng, M. Lops, Y. C. Eldar, and X. Wang, "Radar and communication coexistence: An overview: A review of recent methods," *IEEE Signal Processing Magazine*, vol. 36, no. 5, pp. 85–99, 2019.
- [106] K. V. Mishra, A. Zhitnikov, and Y. C. Eldar, "Spectrum sharing solution for automotive radar," in *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, IEEE, Jun. 2017. DOI: [10.1109/vtcspring.2017.8108413](https://doi.org/10.1109/vtcspring.2017.8108413). [Online]. Available: <https://doi.org/10.1109/vtcspring.2017.8108413>.
- [107] X.-P. Zhu, M. Jin, W.-Q. Qian, S. Liu, and Y.-M. Wei, "The application of unsupervised clustering in radar signal preselection based on doa parameters," in *2010 First International Conference on Pervasive Computing, Signal Processing and Applications*, IEEE, 2010, pp. 956–959.
- [108] Y. Sheng, C. Hou, and W. Si, "Extract pulse clustering in radar signal sorting," in *2017 International Applied Computational Electromagnetics Society Symposium-Italy (ACES)*, IEEE, 2017, pp. 1–2.
- [109] M. Keshavarzi and A. M. Pezeshk, "A simple geometrical approach for deinterleaving radar pulse trains," in *2016 UKSim-AMSS 18th International Conference on Computer Modelling and Simulation (UKSim)*, IEEE, 2016, pp. 172–177.
- [110] K. Gençol, "A two-stage deinterleaving technique for clustering of radar pulses," in *2017 25th Signal Processing and Communications Applications Conference (SIU)*, IEEE, 2017, pp. 1–4.
- [111] A. Ata'a and S. Abdullah, "Deinterleaving of radar signals and prf identification algorithms," *IET radar, sonar & navigation*, vol. 1, no. 5, pp. 340–347, 2007.

- [112] D. Milojević and B. Popović, “Improved algorithm for the deinterleaving of radar pulses,” in *IEE Proceedings F (Radar and Signal Processing)*, IET, vol. 139, 1992, pp. 98–104.
- [113] F. Paisana, J. P. Miranda, N. Marchetti, and L. A. DaSilva, “Database-aided sensing for radar bands,” in *2014 IEEE International Symposium on Dynamic Spectrum Access Networks (DYSPAN)*, IEEE, 2014, pp. 1–6.
- [114] Q. Zhang, H. Wang, Y. Yang, and W. Wang, “Rfcm clustering algorithm based on adaptive weights for radar signal sorting,” in *2016 4th International Conference on Mechanical Materials and Manufacturing Engineering*, Atlantis Press, 2016.
- [115] Q. Guo, Z. Li, and X. Zhang, “Support vector clustering and type-entropy based joint de-interleaving/recognition system of radar pulse sequence,” in *Intelligent Control and Automation*, Springer, 2006, pp. 642–653.
- [116] S.-q. Wang, C. Gao, T. Xu, H.-y. Zeng, and J. Bai, “Radar signal sorting based on core cluster support vector clustering,” in *Recent Trends in Intelligent Computing, Communication and Devices*, Springer, 2020, pp. 859–864.
- [117] K. Gençol, A. Kara, and N. At, “Improvements on deinterleaving of radar pulses in dynamically varying signal environments,” *Digital Signal Processing*, vol. 69, pp. 86–93, 2017.
- [118] S. Bailie and M. Leeser, “Incremental clustering applied to radar deinterleaving: A parameterized fpga implementation,” in *Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays*, 2012, pp. 25–28.

- [119] J. Dudczyk, “Radar emission sources identification based on hierarchical agglomerative clustering for large data sets,” *Journal of Sensors*, vol. 2016, 2016.
- [120] C. Ebersole, A. Buchenroth, D. Zilz, and V. Chakravarthy, “Spatiotemporal density-based clustering for dynamic spectrum sensing,” in *2020 IEEE Radar Conference*, IEEE, 2020.
- [121] G. Revillon, A. Mohammad-Djafari, and C. Enderli, “Radar emitters classification and clustering with a scale mixture of normal distributions,” *IET Radar, Sonar & Navigation*, vol. 13, no. 1, pp. 128–138, 2018.
- [122] G. Revillon and A. Mohammad-Djafari, “A complete classification and clustering model to account for continuous and categorical data in presence of missing values and outliers,” in *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 33, 2019, p. 23.
- [123] Y.-Y. Dong, C.-X. Dong, W. Liu, M.-M. Liu, and Z.-Z. Tang, “Scaling transform based information geometry method for doa estimation,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 6, pp. 3640–3650, 2019.
- [124] R. Q. Quiroga, “Spike sorting,” *Scholarpedia*, vol. 2, no. 12, p. 3583, 2007, revision #137442. DOI: [10.4249/scholarpedia.3583](https://doi.org/10.4249/scholarpedia.3583).
- [125] H. G. Rey, C. Pedreira, and R. Q. Quiroga, “Past, present and future of spike sorting techniques,” *Brain research bulletin*, vol. 119, pp. 106–117, 2015.
- [126] F. Wood, S. Goldwater, and M. J. Black, “A non-parametric bayesian approach to spike sorting,” in *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, IEEE, Aug. 2006. DOI: [10.1109/iembs.2006](https://doi.org/10.1109/iembs.2006).

260700. [Online]. Available: <https://doi.org/10.1109/iembs.2006.260700>.

- [127] J. Gasthaus, F. Wood, D. Gorur, and Y. W. Teh, “Dependent dirichlet process spike sorting,” in *Advances in neural information processing systems*, 2009, pp. 497–504.
- [128] Z. van Havre, N. White, J. Rousseau, and K. Mengersen, “Clustering action potential spikes: Insights on the use of overfitted finite mixture models and dirichlet process mixture models,” *arXiv preprint arXiv:1602.01915*, 2016.
- [129] S. Knieling, K. S. Sridharan, P. Belardinelli, G. Naros, D. Weiss, F. Mormann, and A. Gharabaghi, “An unsupervised online spike-sorting framework,” *International Journal of Neural Systems*, vol. 26, no. 05, p. 1550042, Aug. 2016. DOI: [10.1142/s0129065715500422](https://doi.org/10.1142/s0129065715500422). [Online]. Available: <https://doi.org/10.1142/s0129065715500422>.
- [130] M. Bernert and B. Yvert, “Fully unsupervised online spike sorting based on an artificial spiking neural network,” *BioRxiv*, p. 236224, 2017.
- [131] P. Yger, G. L. Spampinato, E. Esposito, B. Lefebvre, S. Deny, C. Gardella, M. Stimberg, F. Jetter, G. Zeck, S. Picaud, *et al.*, “A spike sorting toolbox for up to thousands of electrodes validated with ground truth recordings in vitro and in vivo,” *Elife*, vol. 7, e34518, 2018.
- [132] J. Wouters, F. Kloosterman, and A. Bertrand, “Towards online spike sorting for high-density neural probes using discriminative template matching with suppression of interfering spikes,” *Journal of neural engineering*, vol. 15, no. 5, p. 056005, 2018.

- [133] B. C. Souza, V. Lopes-dos-Santos, J. Bacelo, and A. B. L. Tort, “Spike sorting with gaussian mixture models,” *Scientific Reports*, vol. 9, no. 1, Mar. 2019. DOI: [10.1038/s41598-019-39986-6](https://doi.org/10.1038/s41598-019-39986-6). [Online]. Available: <https://doi.org/10.1038/s41598-019-39986-6>.
- [134] J. Wild, Z. Prekopcsak, T. Sieger, D. Novak, and R. Jech, “Performance comparison of extracellular spike sorting algorithms for single-channel recordings,” *Journal of neuroscience methods*, vol. 203, no. 2, pp. 369–376, 2012.
- [135] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, “Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering,” *Neural computation*, vol. 16, no. 8, pp. 1661–1687, 2004.
- [136] S. G. Mallat, “A theory for multiresolution signal decomposition: The wavelet representation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 11, no. 7, pp. 674–693, 1989.
- [137] W. M. Mendenhall and T. L. Sincich, *Statistics for Engineering and the Sciences*. CRC Press, 2016.
- [138] C. Rossant, S. N. Kadir, D. F. Goodman, J. Schulman, M. L. Hunter, A. B. Saleem, A. Grosmark, M. Belluscio, G. H. Denfield, A. S. Ecker, *et al.*, “Spike sorting for large, dense electrode arrays,” *Nature neuroscience*, vol. 19, no. 4, p. 634, 2016.
- [139] A. Ahmad and S. S. Khan, “Survey of state-of-the-art mixed data clustering algorithms,” *IEEE Access*, vol. 7, pp. 31 883–31 902, 2019.

- [140] G. Papageorgiou, S. Richardson, and N. Best, “Bayesian non-parametric models for spatially indexed data of mixed type,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 77, no. 5, pp. 973–999, 2015.
- [141] C. Jha, “A nonparametric bayesian method for clustering of high-dimensional mixed dataset,” *arXiv preprint arXiv:1808.04045*, 2018.
- [142] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.
- [143] D. Dua and C. Graff, *UCI machine learning repository*, 2017. [Online]. Available: <http://archive.ics.uci.edu/ml>.
- [144] J. L. Hendrickson, “Methods for clustering mixed data,” Ph.D. dissertation, University of South Carolina, 2014.
- [145] M. S. Shotwell, E. H. Slate, *et al.*, “Bayesian outlier detection with dirichlet process mixtures,” *Bayesian Analysis*, vol. 6, no. 4, pp. 665–690, 2011.
- [146] K. Evans, T. Love, and S. W. Thurston, “Outlier identification in model-based cluster analysis,” *Journal of classification*, vol. 32, no. 1, pp. 63–84, 2015.
- [147] P. Melchior and A. D. Goulding, “Filling the gaps: Gaussian mixture models from noisy, truncated or incomplete samples,” *Astronomy and computing*, vol. 25, pp. 183–194, 2018.
- [148] S. Särkkä, *Bayesian filtering and smoothing*. Cambridge University Press, 2013, vol. 3.