Model-based rare category detection for temporal data

by

Jeroen Mandersloot

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday July 20, 2018 at 14:30 PM.

Student number:	4051874	
Project duration:	November 27, 2017 – July 20, 2018	
Thesis committee:	Dr. D. M. J. Tax,	TU Delft, supervisor
	Prof. M. J. T. Reinders,	TU Delft
	Dr. M. M. de Weerdt,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft Delft University of Technology

This is for you, Dad.

Acknowledgements

First and foremost I would like to thank my supervisor, Dr. David Tax, for his generous guidance. His door and e-mail client were always open for any questions. Throughout my research he kept me focused and organized, while still allowing me complete freedom in my work. His wealth of knowledge and witty sense of humour make him an absolute joy to work with.

I would also like to thank my colleagues in the Pattern Recognition research group for their useful feedback and insightful discussions. I am especially grateful to Madelon. Our frequent conversations and coffee breaks helped me elevate my research, and made the entire process more fun. Her unwavering enthusiasm is an inspiration.

Finally, none of this would have been possible without the incredible support of my wife, Jade. Her continued encouragement these past two years has been invaluable. Whenever stress or anxiety threatened to get the better of me, she was always there to ground me. Thank you.

Contents

Chapter 1

Introduction

Many real-life applications are characterized by highly skewed class proportions. For example, in observational astronomy most of the objects seen in sky surveys are already known: as few as 0.001% actually correspond to new phenomena [13]. This is commonly referred to as class imbalance. Other examples include fraud detection, face recognition, and tumour segmentation. In each of these scenarios the majority of the data consists of "normal" observations (legitimate transactions, two different faces, and healthy tissue), while in fact we are most interested in the minority classes (fraudulent transactions, matching pairs of faces, and malignant tumours).

A lot of research has been conducted to make classification algorithms robust to class imbalance [11], but much of it focuses exclusively on supervised settings. Here, labels for training data are assumed to be readily available, while in practice this may often not be the case. It may not even be known which classes we are looking for. For example, in astronomical sky surveys we are explicitly trying to find unknown phenomena. In such scenarios we first want to discover each of the minority classes present in the data.

The task of discovering all minority classes in an unlabelled dataset is known as rare category detection. It is closely related to anomaly detection, which aims to find individual anomalies. However, there are two important differences. First, in rare category detection we are only interested in minority instances that are representative of an underlying class concept. These instances are therefore assumed to be self-similar. Meanwhile, in anomaly detection the individual anomalies are typically scattered without a clear unifying concept. Second, rare category detection does not aim to identify all minority instances. Instead, the primary goal is to discover the minority classes, so it suffices to find a single instance for each minority class.

Rare category detection is usually an iterative process. During each iteration the algorithm nominates a number of instances that presumably belong to an as of yet undiscovered minority class. A labelling oracle (often a human expert) then provides their true class labels to verify whether this is indeed the case. This process is repeated until all minority classes are discovered. The feedback from the oracle can be used in subsequent iterations to improve the quality of future nominations. Since a labelling oracle is often expensive it is important to minimize the required number of nominations.

There are different approaches to rare category detection, which can be broadly categorized as (1) model-based [28], (2) density-based [12, 13], (3) citation-based [15], or (4) hierarchy-based [36]. Model-based algorithms fit a mixture model to the data and nominate instances that are

poorly explained. Density-based methods assume minority classes are locally compact and look for irregularities in the global density of the data. Citation-based methods try to find regions of interest by considering the number of reverse nearest neighbours of each instance. Hierarchy-based methods build a hierarchy of clusterings and then use a variety of heuristics to decide which clusters likely correspond to minority classes.

Existing research focuses almost exclusively on static data, which is assumed to be independent and identically distributed (i.i.d.). For many real-world problems this may however not be the case. Consider for example video data, where consecutive frames typically depict the same scene. A different example that we use as the main practical motivation throughout this thesis is the detection of bird songs in outdoor audio recordings. Both examples are characterized by strong correlations between consecutive instances. As a result, the distribution of class labels over time is locally smooth.

In this thesis we propose a model-based approach that explicitly incorporates the temporal smoothness of the data. Standard mixture models assume all instances are independent. We present a novel temporal mixture model based on Markov random fields that takes advantage of correlations between instances. We introduce the concept of "temporal neighbourhoods" to describe instances that are close together in time and formulate how these instances affect each other. Compared to static mixture models our temporal model has two distinct advantages: (1) it uses temporal information to improve the model parameter estimates, and (2) it is able to correctly classify instances even when they overlap with another class.

While this thesis focuses on rare category detection, our temporal model is easily applicable to other problem domains. For example, in traditional unsupervised settings without an emphasis on class imbalance it can be used for clustering. In supervised settings our model can be turned into a classifier by adopting a straightforward classification rule. We also give a broadly applicable semi-supervised formulation of our model for partially labelled datasets. These naturally arise in rare category detection as we continually receive new class labels from the oracle.

The contributions of this thesis are therefore twofold. First, we propose a novel formulation for a temporal mixture model based on Markov random fields. We demonstrate several advantages over static mixture models. Second, we present a new model-based rare category detection algorithm. Both these contributions are useful independent of each other. We already explained how our temporal model can be applied outside of rare category detection. Furthermore, out algorithm is model-agnostic so that a different model could easily be substituted for our temporal model.

This thesis is structured as follows. First, we introduce the notation we use throughout this thesis and give a formal problem definition in Chapter 2. Next, we review existing research into rare category detection in Chapter 3. Afterwards, we give a formal introduction of mixture models in Chapter 4. In Chapter 5 we present our temporal model and compare it to static models. Chapter 6 describes our model-based rare category detection algorithm in detail. We evaluate our algorithm based on experiments on several bird song detection tasks in Chapter 7. Then, we discuss the limitations of our work and propose relevant future research directions in Chapter 8. Finally, we conclude our thesis in Chapter 9.

Chapter 2

Preliminaries

We first give an overview of the notation and terminology that we use throughout this thesis. Individual chapters may expand on this to include their own specific content, but common conventions and terms that are relevant for all chapters are presented here. Afterwards, we give a formal definition of the rare category detection problem.

2.1 Notation

At an abstract level we try to adhere to the following style conventions. We denote vectors such as $\mathbf{a}$ by lower case bold Roman letters. Unless otherwise stated we assume all vectors to be column vectors such that

$$\mathbf{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_D \end{bmatrix}. \tag{2.1}$$

Sets and matrices are denoted by upper case bold Roman letters, such as $\mathbf{A}$. Scalar, vector and matrix spaces representing all possible values such quantities could take on are typically denoted by upper case stylized letters like $\mathcal{A}$. We use upper case Roman letters for integers that represent sums or totals such as the dimensionality D of a vector. Integers that are used as indices are written as lower case Roman letters, e.g. $\mathbf{a}_i$ denotes the ith entry of the vector $\mathbf{a}$.

Using the above conventions we can represent data in a variety of ways. We denote an entire dataset by the set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ where $\mathbf{x}_n$ is a D-dimensional feature vector of an individual instance and the size of the dataset is given by N. We could also represent our data as an $N \times D$ matrix

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \ldots & \mathbf{x}_N \end{bmatrix}^{\mathrm{T}}, \tag{2.2}$$

where the superscript T denotes the transpose and the nth row now corresponds to the feature vector of the nth instance. While the symbols for set and matrix notation are identical we make it clear from context which notation we are using if the distinction is relevant. We sometimes use set and matrix notation interchangeably in which case the correspondence between them is analogous to the case of (2.2), i.e. each row in the matrix is a single column vector element in the set.

Each instance $\mathbf{x}_n$ is presumed to have an associated class label y_n. The set of all instance labels is denoted by $\mathbf{Y} = \{y_1, \ldots, y_N\}$, whereas we denote the set of all possible classes by $\mathcal{Y} = \{\omega_1, \ldots, \omega_M\}$

such that each $y_n \in \mathcal{Y}$. In binary classification settings we only have $M = 2$ classes which we typically denote by $\mathcal{Y} = \{-1, +1\}$. Since we focus mainly on problem settings with class imbalance we adhere to the convention that "negative" and "positive" classes correspond to majority and minority classes, respectively. In multi-class scenarios we generalize this notion and define $M = M^- + M^+$, where M^- denotes the number of negative classes and M^+ the number of positive classes.[1] The set of all possible classes is then defined as $\mathcal{Y} = \{-M^-, \ldots, -1, +1, \ldots, +M^+\}$.

We frequently discuss graphs and graphical models, so it is helpful to establish some terminology for them. In this context we deviate slightly from our convention of denoting sets in bold to conform to standard graph notation. Let $G = (V, E)$ denote a graph where V is the set of all nodes and E is the set of all edges. Each edge is a pair of nodes (u, v) with $u, v \in V$ and denotes the existence of a connection from u to v. For undirected graphs it holds that $(u, v) \in E \iff (v, u) \in E$ since there is no concept of directionality. A path $P = \{e_1, e_2, \ldots, e_{|P|}\}$ is an ordered set of edges that connect a sequence of nodes. We define cycles as paths where the first and last node are the same, i.e. $u_1 = v_{|P|}$.

2.2 Problem definition

We now formally define the rare category detection problem. We are given an initially unlabelled $N \times D$ dataset $\mathbf{X}$ and assume we have access to a labelling oracle. The oracle can provide us with the class label of any instance we present it. The task is to find at least one instance from each minority class in $\mathcal{Y}$ using as few labelling requests as possible. This is done by nominating instances that likely belong to a new minority class and querying the oracle for verification.

Rare category detection algorithms typically make some assumptions on the data. These are used to formulate heuristics that decide which instances likely belong to minority classes. The most common assumptions along with a short definition are listed below.

Smoothness: the distribution of the majority class is locally smooth on each dimension.
Compactness: the minority classes form compact clusters in feature space.
Isolation: the minority classes are separable from majority classes in feature space.

While the smoothness and compactness assumptions are adopted by virtually all rare category detection algorithms, some are designed to work specifically in the non-separable case. Individual algorithms may make additional assumptions, and their degree of sensitivity to each assumption may differ as well. We discuss the assumptions for existing algorithms in more detail in Chapter 3.

In this thesis we focus specifically on temporal data rather than static data. In particular, we are interested in data for which the class labels of instances are temporally smooth. We call this the temporal smoothness assumption. Intuitively, this means that consecutive instances that form a "temporal neighbourhood" together generally have a higher probability of belonging to the same class. It is usually straightforward to determine whether a given dataset satisfies this assumption. For example, in video data it is reasonable to assume that consecutive frames depict the same scene and therefore belong to the same class.

[1] Without loss of generality we often assume there is only a single negative class such that $M^- = 1$. If $M^- > 1$ we could always redefine the problem such that all negative classes share the same label.

Chapter 3

Related work

In this chapter we give an overview of existing rare category detection algorithms. We distinguish between (1) model-based, (2) density-based, (3) citation-based, and (4) hierarchy-based algorithms. For each approach we explain the general concept, then describe a number of relevant algorithms, and finally discuss their strengths and weaknesses. The majority of existing algorithms focus exclusively on static data. In the final section of this chapter we cover a few temporal algorithms.

Table 3.1 gives an overview of the algorithms we discuss in this chapter. It also lists some additional properties such as the assumptions on the data, required prior knowledge, and time complexity [20]. These highlight some of the strengths and weaknesses of each algorithm. Regarding the assumptions we distinguish between smoothness (S), compactness (C), and isolation (I) as described in Section 2.2. We use p_m to denote the prior probability of a class ω_m with $1 \leq m \leq M$.

Rare category detection is closely related to anomaly detection, but there are some important differences. In rare category detection we aim to find instances that are representative of minority classes and therefore self-similar. Also, we are mainly interested in discovering all minority classes rather than identifying individual anomalous instances. While a comprehensive review of anomaly detection is outside the scope of this thesis, we refer the curious reader to [7, 26].

Table 3.1: An overview of existing rare category detection algorithms.

	Algorithm	**Category**	**Assumptions**	**Prior knowledge**	**Time complexity**
Static	Interleave [28]	model-based	S, C, I	M	$\mathcal{O}(DN^2)$
	NNDM [13]	density-based	S, C	$M, p_1, \ldots, p_M$	$\mathcal{O}(DN^{2-\frac{1}{D}})$
	GRADE [14]	density-based	S, C	$M, p_1, \ldots, p_M$	$\mathcal{O}(DN^3)$
	SEDER [12]	density-based	S, C	-	$\mathcal{O}(D^2N^2)$
	FRED [20]	density-based	S, C	-	$\mathcal{O}(DN)$
	RADAR [16]	citation-based	S, C	$M, p_1, \ldots, p_M$	$\mathcal{O}(DN^2)$
	CLOVER [15]	citation-based	S, C, I	-	$\mathcal{O}(DN^{2-\frac{1}{D}})$
	HMS [36]	hierarchy-based	S, C, I	-	$\mathcal{O}(DN^2)$
Temporal	BIRAD [40]	model-based	S, C, $M = 2$[1]	p_1, p_2	$\mathcal{O}(DN)$
	BIRD [41]	density-based	S, C	$M, p_1, \ldots, p_M$	$\mathcal{O}(DN^2)$

[1] BIRAD is only applicable to binary problems where $M = 2$. Additionally, BIRAD uses hidden Markov models and is therefore also dependent on the Markov assumption. We briefly cover hidden Markov models in Chapter 5.

3.1 Model-based algorithms

Model-based algorithms fit a mixture model to the data and identify instances that are poorly explained. Each component usually follows a Gaussian distribution, but other distributions can be used as well. Because the dataset as a whole is dominated by the majority classes it is expected that instances from the minority classes are poorly explained by this model. Instances that do not fit the model well are therefore considered more likely to belong to a minority class and are nominated first. After each iteration the model is improved based on the feedback from the oracle before nominating the next batch of instances. Figure 3.1 visualizes the iterative scheme adopted by model-based algorithms.

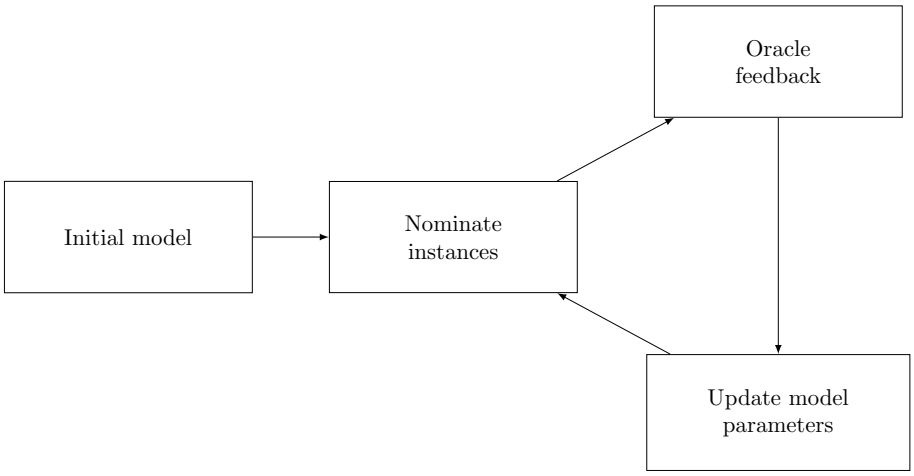


Figure 3.1: The iterative scheme for model-based rare category detection algorithms.

An example of a model-based algorithm is Interleave [28]. It models each class using a single Gaussian component and therefore requires the total number of classes M as prior knowledge. After the model is fitted to the data each instance is assigned to the component that explains it best, i.e. assigns the highest likelihood to it. Each component then nominates the most anomalous instance assigned to it. This strategy is different from nominating instances based on lowest overall likelihood according to the model as a whole. Importantly, it eliminates the mixture weights from the equation.[2] This prevents components with relatively large mixture weights from dominating the nomination process.

An advantage of model-based algorithms is that they can be applied to a wide variety of datasets. They are often model-agnostic, so different distributions can be used without requiring further adaptations. By introducing additional components we can increase the model complexity to accurately model even highly irregular datasets. This highlights a limitation of the Interleave algorithm, which models each class using a single Gaussian. A downside of model-based algorithms in general is that many require classes to be separable in feature space. If a minority class overlaps completely with one of the majority classes it is difficult for a mixture model to accurately describe them.

[2] In Chapter 4 we formally introduce mixture models and explain mixture weights in more detail.

3.2 Density-based algorithms

Density-based algorithms estimate the local density near each instance and repeatedly nominate instances from regions with a maximum change of local density [12, 13]. As such, density-based methods are also applicable in the non-separable case. This strategy is strongly dependent on the smoothness and compactness assumptions. If these assumptions hold, regions that contain minority classes should be clearly distinguishable. An example is shown in Figure 3.2.

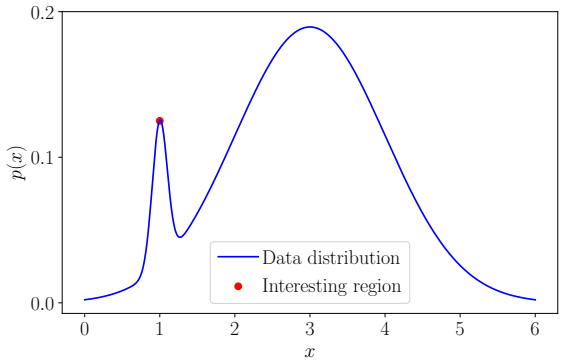


Figure 3.2: An example of a data distribution where the minority class overlaps with the majority class.

A common approach for estimating local densities is by considering the number of nearest neighbours for each instance. If the distribution of the majority class is indeed smooth then this number also varies smoothly. Instances for which this number changes suddenly may indicate the presence of a minority class and are therefore nominated. This approach is adopted by both NNDM [13] and GRADE [14]. The difference is that GRADE also takes into account global similarities between instances. This allows GRADE to exploit potential manifold structures in the data.

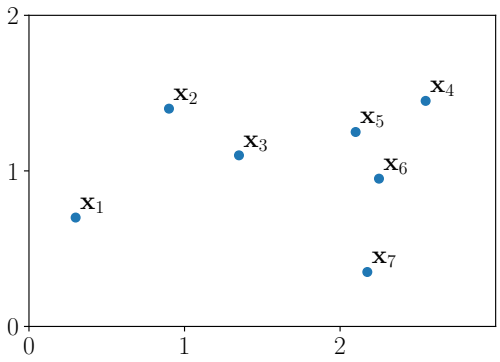
SEDER [12] is a prior-free algorithm that performs semi-parametric density estimation to obtain a measure of the local densities. It first applies kernel density estimation using Gaussian kernels. Since the resulting density function is often under-smoothed, the smoothness assumption would generally not hold. This is solved by introducing additional data-dependent smoothing terms whose parameters are learned through maximum likelihood estimation.

FRED [20] initially groups the instances into separate bins on each dimension of the data. Each bin thus represents a small region in feature space. Afterwards, the local density for each bin is estimated through histogram density estimation. The resulting densities can then be interpreted as a function of the bins. By conducting wavelet analysis on these density functions the regions corresponding to sudden changes in local density can be found.

An advantage of density-based methods is that they can be used even when minority classes overlap completely with the majority class. However, this strategy only works if the minority classes have significantly higher local density than the majority class [16]. This makes density-based algorithms more reliant on the smoothness and compactness assumptions. Also, it is generally more difficult for density-based methods to incorporate labelled feedback from the oracle to improve subsequent nominations.

3.3 Citation-based algorithms

Citation-based methods use nearest neighbours to find minority classes and are therefore somewhat similar to density-based algorithms like NNDM and GRADE. An important difference is that citation-based methods also take into account reverse nearest neighbours (RNNs). Instead of only considering the K nearest neighbours for a given instance $\mathbf{x}_n$, they also consider instances for which $\mathbf{x}_n$ is itself a nearest neighbour. Figure 3.3 shows a simple scatter plot along with all (reverse) nearest neighbours for $K = 2$.



Instance	NNs	RNNs
$\mathbf{x}_1$	$\mathbf{x}_2, \mathbf{x}_3$	$\mathbf{x}_2$
$\mathbf{x}_2$	$\mathbf{x}_1, \mathbf{x}_3$	$\mathbf{x}_1, \mathbf{x}_3$
$\mathbf{x}_3$	$\mathbf{x}_2, \mathbf{x}_5$	$\mathbf{x}_1, \mathbf{x}_2$
$\mathbf{x}_4$	$\mathbf{x}_5, \mathbf{x}_6$	$\mathbf{x}_5, \mathbf{x}_6$
$\mathbf{x}_5$	$\mathbf{x}_4, \mathbf{x}_6$	$\mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_6, \mathbf{x}_7$
$\mathbf{x}_6$	$\mathbf{x}_4, \mathbf{x}_5$	$\mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_7$
$\mathbf{x}_7$	$\mathbf{x}_5, \mathbf{x}_6$	-

Figure 3.3: The (reverse) nearest neighbours in a simple 2D example for $K = 2$.

RADAR uses the number of RNNs to identify minority instances on the boundary with a majority class [16]. The authors distinguish these boundary instances from "inner instances", which are closer to the cluster centres. Inner instances are typically grouped more densely and therefore expected to have more RNNs. By the same logic, the nearest neighbours of inner instances are almost exclusively other inner instances. Boundary instances mostly neighbour instances far from the cluster centre. For each instance RADAR considers the ratio between the average and the variance of the number of RNNs among its nearest neighbours. It is expected that inner instances have a higher average and lower variance, while the opposite holds for boundary instances.

CLOVER looks at mutual nearest neighbours instead of reverse nearest neighbours. It is designed to differentiate between uninteresting isolated anomalies and the clustered minority instances we are actually interested in. The intuition is that isolated anomalies have almost no mutual nearest neighbours, while the self-similar instances in minority classes do. To distinguish potential minority instances from majority instances the algorithm considers the distance to the Kth nearest neighbour. If the minority classes are isolated it is expected that this distance drastically increases for minority instances once K is larger than the size of their class, while it remains relatively stable for majority instances.

In general, citation-based algorithms are used to distinguish interesting instances based on their neighbourhood. A weakness these algorithms have in common is the need to estimate a hyperparameter K that specifies the number of nearest neighbours to consider. This could be based partly on prior knowledge as in RADAR, but this is not the case for CLOVER. Additionally, like density-based algorithms, it is often not straightforward how to incorporate oracle feedback.

3.4 Hierarchy-based algorithms

Hierarchy-based algorithms repeatedly cluster the data on different hierarchical levels. Clusters that likely correspond to minority classes are identified by analysing the clusterings across various levels of the hierarchy. Each cluster is assigned a score based on a validity criterion which measures certain properties that may be indicative of a minority class. Once each cluster is assigned a score the algorithm nominates the centre-most instance from the cluster with the highest score.

Examples of validity criteria are "outlierness" and "compactness-isolation" [36]. Outlierness measures the ratio between the lifetime and size of a cluster. It is expected that anomalous clusters survive for a long time in the hierarchy, since they are relatively far away from other clusters. Similarly, smaller clusters are also more likely to be anomalous. For anomalous clusters this ratio therefore tends to be large. Compactness-isolation measures the quality of a cluster. Intuitively, a cluster can be considered "good" if the distance to the centre is small for instances inside (compactness) and large for instances outside (isolation).

HMS is currently the only existing hierarchy-based algorithm [36]. It uses the mean shift algorithm with increasing bandwidths to repeatedly cluster the data. This creates a cluster hierarchy where each bandwidth corresponds to a single level. The authors show that the compactness-isolation criterion generally yields better results than the outlierness criterion. It may happen that multiple clusters share the same criterion score. The authors propose to resolve such ties using a highest average distance (HAD) heuristic. Specifically, they prioritize the cluster with the highest average distance from its centre-most instance to all previously labelled instances.

The performance of hierarchy-based methods depends largely on the validity criterion and clustering algorithm that are used. This makes hierarchy-based methods more flexible since an appropriate clustering algorithm can be chosen depending on the data. Some algorithms require hyperparameters to be specified manually, such as the bandwidth in mean shift. Hierarchy-based methods that use hard-partitioning clustering algorithms require the minority classes to be separable in feature space. This requirement may be relaxed somewhat for fuzzy clustering algorithms, but even then it remains infeasible to discover minority classes that overlap completely with the majority class.

3.5 Temporal algorithms

Existing research for rare category detection on temporal data is extremely limited. We are aware of only two algorithms that explicitly incorporate temporal aspects of the data. However, neither is particularly suitable for the kind of temporal data that we are interested in.

BIRAD [40] exploits bi-level temporal structures in the data. Given several temporal sequences it aims to label both the sequences as a whole and the individual time segments within them. Consider for example ECG signals collected mostly from healthy people and a small number of patients suffering from a heart disease. Most signals will be normal, and even within the anomalous signals only a few segments will show abnormal patterns. The applicability of BIRAD is limited to binary problems since it cannot distinguish between different minority classes.

BIRD [41] is designed to work specifically on time-evolving graphs. Such graphs are useful when the minority classes themselves change over time. This may for example be the case in virtual identity detection. Here, a criminal may intentionally change their fraudulent behaviour over time to avoid being caught.

Chapter 4

Mixture models

We begin the discussion of our model-based approach by reviewing the concepts underpinning mixture models in general. In later sections we build upon the concepts introduced here. By doing so, we also make this thesis self-sufficient in providing the reader with all the necessary background information. First, we give a formal introduction to mixture models and define their probability densities, likelihood functions, and other properties. Afterwards, we cover the expectation-maximization algorithm that is widely used for computing maximum likelihood estimates of the model parameters. Finally, we discuss how to use partial label information to improve our model through semi-supervised learning.

4.1 Definitions

Mixture models are defined by a linear combination of more basic probabilistic models. As such they are capable of modelling more complex densities than their individual components. We can express any mixture model with K components as

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k p_k(\mathbf{x}|\boldsymbol{\theta}_k), \tag{4.1}$$

where π_k is the mixture weight of the kth component and $p_k(\mathbf{x}|\boldsymbol{\theta}_k)$ is the corresponding probability density function parametrized by $\boldsymbol{\theta}_k$. In order for $p(\mathbf{x})$ to be a valid probability density function the mixture weights $\boldsymbol{\pi}$ have to satisfy

$$\sum_{k=1}^{K} \pi_k = 1 \tag{4.2}$$

and

$$0 \leq \pi_k \leq 1 \tag{4.3}$$

for all $k = 1, \ldots, K$.

We use the subscript k for the density functions $p_k(\mathbf{x}|\boldsymbol{\theta}_k)$ because in general the mixture model may be composed of components with functionally different distributions. However, in practice we often define mixture models as combinations of a single type of distribution. In this thesis we focus

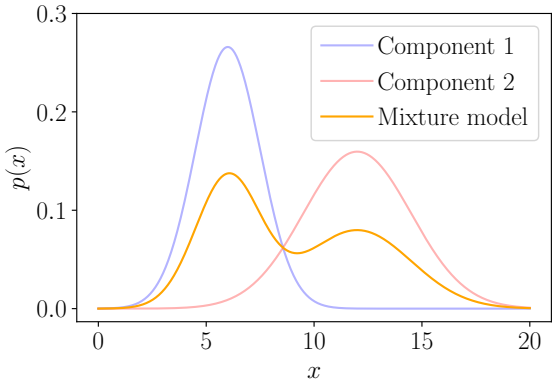


Figure 4.1: The probability density functions of a Gaussian mixture model and its two individual components.

on Gaussian mixture models where each component follows a Gaussian distribution. The subscript is then unnecessary since the functional forms of the density functions are identical (though their individual parameters $\boldsymbol{\theta}_k$ still vary). We can then write (4.1) as

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{4.4}$$

where

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_k|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^{\mathrm{T}}\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right) \tag{4.5}$$

denotes the Gaussian distribution of the kth component parametrized by the mean vector $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$.

Figure 4.1 shows the probability density functions of a univariate Gaussian mixture model and its two individual Gaussian components. The mixture weights of both components are identical so that $\pi_1 = \pi_2 = 0.5$, which explains why the peaks in the mixture model are roughly halved in size. It is clear that by adding more components and tuning the mixture weights we can model increasingly more complex densities.

A useful way to think about mixture models is through discrete latent variables (also called hidden or unobserved variables) that indicate which component generated a particular instance. Specifically, for each instance $\mathbf{x}_n$ we define a corresponding K-dimensional binary random variable $\mathbf{z}_n$ having a 1-of-K representation. That is, all entries of $\mathbf{z}_n$ equal 0 except for the kth entry which equals 1. If $z_{nk} = 1$ we say $\mathbf{x}_n$ belongs to component k. The $N \times K$ matrix containing all latent variables is given by

$$\mathbf{Z} = \begin{bmatrix}\mathbf{z}_1 & \dots & \mathbf{z}_N\end{bmatrix}^{\mathrm{T}}, \tag{4.6}$$

in which each row corresponds to a single latent variable $\mathbf{z}_n$. For now we still assume all instances are independent, so we drop the subscript n for notational clarity.

This formulation is equivalent to (4.4) except we now explicitly assign a probabilistic meaning to the mixture weights $\boldsymbol{\pi}$. This was already hinted at by (4.2) and (4.3), which are sufficient conditions

for $\boldsymbol{\pi}$ to be a valid probability vector. To show this, let $\mathbf{z}$ follow a categorical distribution (which is just a special case of the multinomial distribution with a single trial). By definition we then have

$$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}, \tag{4.7}$$

or equivalently

$$p(z_k = 1) = \pi_k. \tag{4.8}$$

Since the non-zero entry of $\mathbf{z}$ indicates which component $\mathbf{x}$ belongs to we have

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k \boldsymbol{\Sigma}_k)^{z_k}. \tag{4.9}$$

Using the same notation as in (4.8) we can also write

$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{4.10}$$

Consider now the joint distribution $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$. By marginalizing out $\mathbf{z}$ we obtain an expression for $p(\mathbf{x})$ which we can then compare to (4.4). Doing so gives us

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) \tag{4.11}$$

$$= \sum_{k=1}^{K} p(z_k = 1)p(\mathbf{x}|z_k = 1) \tag{4.12}$$

$$= \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{4.13}$$

which is indeed identical to our previous expression.

A graphical representation of this probabilistic interpretation is shown in Figure 4.2. Here we show explicitly how the random variables $\mathbf{x}$ and $\mathbf{z}$ are influenced by their parameters and each other. We include the subscript n in the nodes for $\mathbf{x}_n$ and $\mathbf{z}_n$ inside the rectangular plate to indicate there are N such variables in total. The node for $\mathbf{x}_n$ is shaded to indicate that $\mathbf{x}_n$ is an observed variable, 5in contrast to $\mathbf{z}_n$.

We now consider the likelihood function $L(\boldsymbol{\Theta})$ given all observed data $\mathbf{X}$ defined as

$$L(\boldsymbol{\Theta}) \equiv p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{4.14}$$

Here, $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K\}$ is the set of all component parameters. We are interested in finding the parameter values $\boldsymbol{\Theta}^*$ for which this likelihood function is maximized, i.e.

$$\boldsymbol{\Theta}^* = \arg\max_{\boldsymbol{\Theta}} L(\boldsymbol{\Theta}). \tag{4.15}$$

Instead of maximizing the likelihood directly we often maximize the log-likelihood. Since the logarithm is a monotonically increasing function the maximum value of $\log L(\boldsymbol{\Theta})$ is achieved for

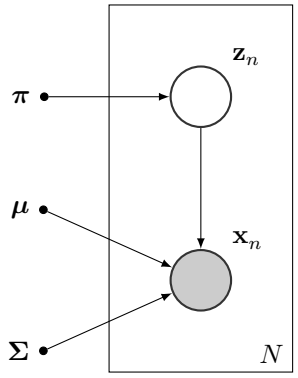


Figure 4.2: A graphical representation of a Gaussian mixture model.

the same value of $\boldsymbol{\Theta}$ that maximizes $L(\boldsymbol{\Theta})$. This not only simplifies the subsequent mathematical analysis but also prevents numerical precision errors associated with multiplying many exponentially small factors together. The log-likelihood function is defined as

$$\begin{aligned}\log L(\boldsymbol{\Theta}) &= \log \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \\ &= \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \end{aligned} \tag{4.16}$$

While (4.16) does replace the product over all N instances with a summation, we still have to deal with the sum inside the logarithm. This is where we can use the latent variables $\mathbf{Z}$ to our advantage. It turns out that we can get rid of the inner summation by considering the joint distribution $p(\mathbf{X}, \mathbf{Z})$. We denote the corresponding log-likelihood function by $\log L_C(\Theta)$ and commonly refer to it as the complete-data log-likelihood. By contrast, $\log L(\boldsymbol{\Theta})$ is called the incomplete-data log-likelihood because it only considers $\mathbf{X}$. Since each $\mathbf{z}$ has a 1-of-K representation we can write

$$\begin{aligned}\log L_C(\boldsymbol{\Theta}) \equiv \log p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) &= \log \prod_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \\ &= \sum_{n=1}^{N} \log \sum_{k=1}^{K} z_{nk} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \\ &= \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \log \left(\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) \\ &= \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \left(\log \pi_k + \log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right), \end{aligned} \tag{4.17}$$

where now the logarithm crucially acts only on a single Gaussian. This is desirable because we know

$$\log \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{1}{2} \log |\boldsymbol{\Sigma}| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) + \text{const}, \tag{4.18}$$

which we can use to further simplify this expression.

The obvious problem with this approach is that in general we do not know the values of the latent variables $\mathbf{Z}$. The only thing we know about them is their posterior distribution $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\Theta})$. To make it clear how we can use $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\Theta})$ in conjunction with (4.17) to estimate $\boldsymbol{\Theta}^*$ we turn to a discussion of the expectation-maximization algorithm.

4.2 Expectation-maximization algorithm

The expectation-maximization (EM) algorith is a widely used method for obtaining maximum likelihood estimates in models with latent variables [25]. It was first proposed in [9] and has since been applied successfully to many different fields [24]. Following the approach of [5] and Chapter 9.3 of [4] we mainly discuss the EM algorithm from a practical perspective. While a more fundamental discussion of its theoretical properties and guarantees is also interesting, it is not directly relevant for our thesis. We refer the curious reader to the original EM paper or Chapter 9.4 of [4] for a more abstract treatment.

The EM algorithm is initialized with a "guess" of the model parameters $\boldsymbol{\Theta}^{(0)}$ and then iteratively updates them to obtain $\boldsymbol{\Theta}^{(t+1)}$. Each iteration consists of an E-step (or expectation step) and an M-step (or maximization step), hence giving the algorithm its name. In the E-step we use our current parameter estimates $\boldsymbol{\Theta}^{(t)}$ to compute the posterior probabilities $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\Theta})$. In the M-step we then use these probabilities to improve our estimates and obtain $\boldsymbol{\Theta}^{(t+1)}$. This process is repeated until the change in log-likelihood or parameter estimates falls below a certain threshold. This iterative scheme is necessary because there are generally no closed-form solutions to (4.15) and so we have to approximate it. For the first iteration we need an initial estimate of $\boldsymbol{\Theta}^{(0)}$, which is usually done by running the K-Means algorithm as a pre-processing step [22].

We now give formal definitions of the two separate steps, starting with the E-step. As stated before we have no way of knowing $\mathbf{Z}$; we only know the posterior distribution $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\Theta})$. This means we cannot directly maximize the complete-data log-likelihood given by (4.17). Instead, we consider its expected value denoted by $Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(t)})$ and given by

$$\begin{aligned}
Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(t)}) &= \mathbb{E}_{\mathbf{Z}}\left[\log L_C\left(\boldsymbol{\Theta}\right) \;\middle|\; \mathbf{X}, \boldsymbol{\Theta}^{(t)}\right] \\
&= \mathbb{E}_{\mathbf{Z}}\left[\sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\left(\log \pi_k + \log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) \;\middle|\; \mathbf{X}, \boldsymbol{\Theta}^{(t)}\right] \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{E}[z_{nk}|\mathbf{X}, \boldsymbol{\Theta}^{(t)}]\left(\log \pi_k + \log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} p(z_{nk} = 1|\mathbf{x}_n, \boldsymbol{\Theta}^{(t)})\left(\log \pi_k + \log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right). \qquad (4.19)
\end{aligned}$$

Here, we use linearity of expectation to simplify the expression, and subsequently observe that $\mathbb{E}[z_{nk}|\mathbf{X}, \boldsymbol{\Theta}^{(t)}] = p(z_{nk} = 1|\mathbf{x}_n, \boldsymbol{\Theta}^{(t)})$. This quantity is commonly referred to as the responsibility of component k for instance $\mathbf{x}_n$. As a shorthand we denote it by $\gamma(z_{nk})$ and leave out the dependence

on the parameters for clarity. Using Bayes' theorem we can write

$$\begin{aligned}\gamma(z_{nk}) \equiv p(z_{nk}=1|\mathbf{x}_n) &= \frac{p(z_{nk}=1)p(\mathbf{x}_n|z_{nk}=1)}{p(\mathbf{x}_n)} \\ &= \frac{p(z_{nk}=1)p(\mathbf{x}_n|z_{nk}=1)}{\sum_{j=1}^{K} p(z_{nj}=1)p(\mathbf{x}_n|z_{nj}=1)}.\end{aligned} \tag{4.20}$$

After computing the responsibilities in the E-step we use them to update our parameter estimates $\boldsymbol{\Theta}^{(t+1)}$ for the next iteration. This is done in the M-step of the algorithm, so called because we maximize (4.19) with respect to the parameters $\boldsymbol{\Theta}$. Formally, we have

$$\boldsymbol{\Theta}^{(t+1)} = \arg\max_{\boldsymbol{\Theta}} Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(t)}), \tag{4.21}$$

which we can solve by setting the partial derivatives with respect to each individual parameter $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ and π_k equal to 0. To make sure that in each case we are dealing with a (unique) maximum we have to check the functional form of (4.19) with respect to the parameter we are updating. Writing out the logarithm of the Gaussian distribution using (4.18) and observing that $-\frac{D}{2}\log 2\pi$ is just a constant we obtain

$$Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(t)}) = \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\Big(\log\pi_k - \frac{1}{2}\log|\boldsymbol{\Sigma}_k| - \frac{1}{2}(\mathbf{x}_n-\boldsymbol{\mu}_k)^{\mathrm{T}}\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_n-\boldsymbol{\mu}_k)\Big) + \text{const.} \tag{4.22}$$

It is clear that if we limit ourselves to non-singular covariance matrices $\boldsymbol{\Sigma}_k$, and take into account the constraints (4.2) and (4.3) for π_k, the stationary points that we find do indeed correspond to maxima.

We now derive the update rules for each individual parameter, starting with $\boldsymbol{\mu}_k$. First, we compute the partial derivative given by

$$\frac{\partial Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(t)})}{\partial \boldsymbol{\mu}_k} = -\sum_{n=1}^{N}\gamma(z_{nk})\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_n-\boldsymbol{\mu}_k) \tag{4.23}$$

and set it equal to 0. If we then multiply by $\boldsymbol{\Sigma}_k$ and rewrite in terms of $\boldsymbol{\mu}_k$ we find the update rule

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{\sum_{n=1}^{N}\gamma(z_{nk})\mathbf{x}_n}{\sum_{n=1}^{N}\gamma(z_{nk})}. \tag{4.24}$$

We can do the same for $\boldsymbol{\Sigma}_k$. The partial derivative is then given by

$$\frac{\partial Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(t)})}{\partial \boldsymbol{\Sigma}_k} = -\frac{1}{2}\sum_{n=1}^{N}\gamma(z_{nk})\boldsymbol{\Sigma}_k^{-1} + \frac{1}{2}\sum_{n=1}^{N}\gamma(z_{nk})\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_n-\boldsymbol{\mu}_k)^{\mathrm{T}}(\mathbf{x}_n-\boldsymbol{\mu}_k)\boldsymbol{\Sigma}_k^{-1}, \tag{4.25}$$

which we again set equal to 0. After multiplying by $\boldsymbol{\Sigma}_k$ twice and rewriting we obtain the update rule

$$\boldsymbol{\Sigma}_k^{(t+1)} = \frac{\sum_{n=1}^{N}\gamma(z_{nk})(\mathbf{x}_n-\boldsymbol{\mu}_k)(\mathbf{x}_n-\boldsymbol{\mu}_k)^{\mathrm{T}}}{\sum_{n=1}^{N}\gamma(z_{nk})}. \tag{4.26}$$

Both (4.24) and (4.26) can intuitively be interpreted as the weighted sample mean and covariance, respectively. The weights are then given by the responsibilities $\gamma(z_{nk})$.

For the last parameter π_k we have to take into account the constraint in (4.2). We therefore incorporate a Lagrange multiplier and take the partial derivative of the resulting function given by

$$\frac{\partial\Big(Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(t)}) + \lambda\Big(\sum_{j=1}^{K} \pi_j - 1\Big)\Big)}{\partial \pi_k} = \frac{1}{\pi_k} \sum_{n=1}^{N} \gamma(z_{nk}) + \lambda, \tag{4.27}$$

which we once more set equal to 0. To solve for π_k we first need to determine the value of λ. We can do so by multiplying both sides by π_k and summing the resulting expression over K, allowing us to make use of (4.2). Rewriting tells us that $\lambda = -N$ and so our final update rule is given by

$$\pi_k^{(t+1)} = \frac{1}{N} \sum_{n=1}^{N} \gamma(z_{nk}). \tag{4.28}$$

While we can easily evaluate each of these update rules, they are not actually closed-form solutions because the responsibilities $\gamma(z_{nk})$ themselves also depend on the parameters $\boldsymbol{\Theta}^{(t)}$. We can see this immediately from the definition in (4.20). This again illustrates the need for an iterative update scheme: we first compute $\gamma(z_{nk})$ in the E-step by keeping $\boldsymbol{\Theta}^{(t)}$ fixed, and then compute $\boldsymbol{\Theta}^{(t+1)}$ in the M-step by keeping $\gamma(z_{nk})$ fixed. During the next iteration the new responsibilities are computed using $\boldsymbol{\Theta}^{(t+1)}$, and so on, until the algorithm has converged.

4.3 Semi-supervised learning

In our treatment of the EM algorithm so far we assumed nothing was known about $\mathbf{Z}$ except the posterior distribution $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\Theta})$. However, in rare category detection we can obtain additional information about $\mathbf{Z}$ through a labelling oracle. After nominating an instance we obtain its class label from the oracle for verification. Regardless of whether the instance belongs to a minority or majority class, this additional information can be used to improve the quality of subsequent nominations.

The above suggests that rare category detection bears a close resemblance to active learning [32]. Both make use of a labelling oracle which is used to improve the performance of a system. However, there is an important difference in the motivation for selecting instances that are to be labelled [8]. In active learning the goal is to identify instances that carry the most potential information. These are typically instances for which knowing the class label is expected to improve the performance of a classifier the most. In rare category detection the goal of labelling instances is not to improve our system, but to check whether we have discovered a new minority class. The label information can almost be considered incidental, but since it is given to us we might as well use it.

More generally speaking we now move from unsupervised learning (where we know nothing about the class labels) to semi-supervised learning (where we know some class labels). We can split our data in two disjoint subsets $\mathbf{X}_U, \mathbf{X}_L \subseteq \mathbf{X}$ where $\mathbf{X}_U \cap \mathbf{X}_L = \emptyset$ and $\mathbf{X}_U \cup \mathbf{X}_L = \mathbf{X}$. Initially we have no additional information, so $\mathbf{X}_U = \mathbf{X}$. Each time we then request the class label for an instance $\mathbf{x}_n$ it is moved from $\mathbf{X}_U$ to $\mathbf{X}_L$. At any point we denote the size of $\mathbf{X}_U$ and $\mathbf{X}_L$ by N_U and N_L, respectively. In most semi-supervised settings the unlabelled instances greatly outnumber the labelled instances so that $N_U \gg N_L$.

Analogously, we also split the latent variables $\mathbf{Z}$ into two disjoint subset $\mathbf{Z}_U$ and $\mathbf{Z}_L$. It is important to emphasize that the latent variables do not directly represent the class labels of the instances. Those are instead presumed to be in the set $\mathbf{Y}$ which we have ignored until now. We assume that each class may be modelled by one or more Gaussian components so that $K \geq M$. We only have $K = M$ when each class is modelled by a single Gaussian.

When talking about the components associated with a particular class it is useful to introduce some additional notation. For any class $\omega \in \mathcal{Y}$ we say ω is modelled by the components $\mathbf{C}_\omega \subseteq \{1, \ldots, K\}$. Inversely, we say the components $\mathbf{C}_\omega$ model ω and each $k \in \mathbf{C}_\omega$ is assigned to ω. For notational clarity we write $k \mapsto \omega$ to denote this assignment. Every pair of distinct $\mathbf{C}_{\omega_i}, \mathbf{C}_{\omega_j}$ with $i \neq j$ are mutually disjoint such that $\mathbf{C}_{\omega_i} \cap \mathbf{C}_{\omega_j} = \emptyset$. In other words, each component can only be assigned to a single class. Alternative formulations have been proposed where instead components are soft-assigned to classes [23]. While some performance improvements are reported, we prefer to use the hard-assignment model here to keep the notation uncluttered and the explanation intuitive. It is however trivial to extend our formulation to incorporate soft-assignment.

After observing the class label y_n of an instance $\mathbf{x}_n$ we can now formulate the class-conditional probability as

$$p(z_{nk} = 1|y_n) = \begin{cases} \frac{\pi_k}{\sum_{j \mapsto y_n} \pi_j} & \text{if } k \mapsto y_n \\ 0 & \text{otherwise,} \end{cases} \tag{4.29}$$

where the summation is over all components j assigned to y_n. We can analogously update the definition of $\gamma(z_{nk})$ to obtain the class-conditional responsibilities denoted by $\gamma(z_{nk}|y_n)$ as follows:

$$\begin{aligned} \gamma(z_{nk}|y_n) \equiv p(z_{nk} = 1|\mathbf{x}_n, y_n) &= \frac{p(z_{nk} = 1|y_n)p(\mathbf{x}_n|z_{nk} = 1, y_n)}{\sum_{j=1}^{K} p(z_{nj} = 1|y_n)p(\mathbf{x}_n|z_{nj} = 1, y_n)} \\ &= \frac{p(z_{nk} = 1|y_n)p(\mathbf{x}_n|z_{nk} = 1)}{\sum_{j=1}^{K} p(z_{nj} = 1|y_n)p(\mathbf{x}_n|z_{nj} = 1)} \\ &= \begin{cases} \frac{\gamma(z_{nk})}{\sum_{j \mapsto y_n} \gamma(z_{nj})} & \text{if } k \mapsto y_n \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{4.30}$$

Here, we have made use of the fact that y_n and $\mathbf{x}_n$ are conditionally independent given $\mathbf{z}_n$ and so we know that $p(\mathbf{x}_n|z_{nk} = 1, y_n) = p(\mathbf{x}_n|z_{nk} = 1)$.

Figure 4.3 shows the updated graphical model where we now explicitly incorporate the class labels y_n. We now split the nodes into two subsets depending on whether y_n is hidden or observed. To establish whether two variables are conditionally independent according to a graphical model we can use the d-separation criterion [27]. A full discussion of d-separation is beyond the scope of this thesis, but formal treatments are given in Chapter 3 of [27] and Chapter 8 of [4].

In order to use the observed class labels to improve our parameter estimates we have to reconsider the log-likelihood function. We first update our definitions of the incomplete and complete data, since these now differ for the unlabelled and labelled parts. For the unlabelled part the definitions remain similar in that we consider the incomplete data to be given by $\mathbf{X}_U$ and the complete data by $\{\mathbf{X}_U, \mathbf{Z}_U\}$. For the labelled part we now also observe the class labels so that the incomplete data is given by $\{\mathbf{X}_L, \mathbf{Y}_L\}$ and the complete data by $\{\mathbf{X}_L, \mathbf{Z}_L, \mathbf{Y}_L\}$. These definitions are also shown in

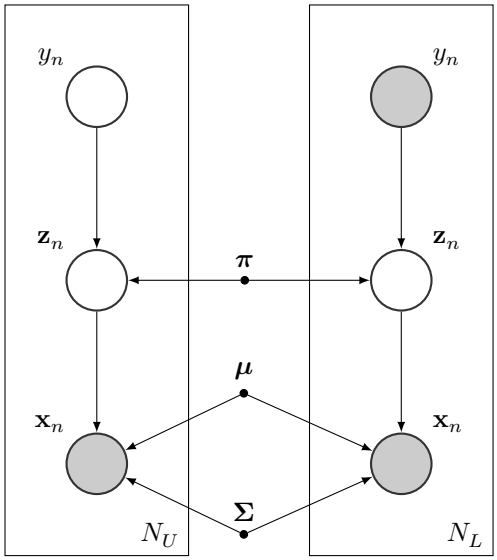


Figure 4.3: A graphical representation of a Gaussian mixture model in a semi-supervised setting.

Table 4.1 for clarity. Note that $\mathbf{Y}_U$ is still ignored because it clutters the notation and not much is gained by incorporating the unknown class labels into our model.

Table 4.1: The incomplete and complete data for the unlabelled and labelled parts.

	Unlabelled	Labelled
Incomplete	$\mathbf{X}_U$	$\mathbf{X}_L, \mathbf{Y}_L$
Complete	$\mathbf{X}_U, \mathbf{Z}_U$	$\mathbf{X}_L, \mathbf{Y}_L, \mathbf{Z}_L$

Next, we update our definition of $Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(t)})$ so we can derive new update rules that incorporate $\mathbf{Y}_L$. For this we first observe that the posterior probability $p(y_n|z_{nk} = 1)$ is readily given by

$$p(y_n|z_{nk} = 1) = \begin{cases} 1 & \text{if } k \mapsto y_n \\ 0 & \text{otherwise,} \end{cases} \tag{4.31}$$

meaning y_n is fully determined once we know $\mathbf{z}_n$. As a result, the definition of the complete-data log-likelihood $\log L_C(\boldsymbol{\Theta})$ given by (4.17) does not change, since $\mathbf{Y}_L$ contains no new information if we already know $\mathbf{Z}$. However, its expected value $Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(t)})$ is affected by $\mathbf{Y}_L$ because we do not actually know $\mathbf{Z}$. By first splitting (4.17) into two separate summations over the unlabelled and

labelled data, respectively, and then taking its expected value we obtain

$$\begin{aligned}
Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(t)}) &= \mathbb{E}_{\mathbf{Z}}\left[\log L_C\left(\boldsymbol{\Theta}\right) \;\middle|\; \mathbf{X}_U, \mathbf{X}_L, \mathbf{Y}_L, \boldsymbol{\Theta}^{(t)}\right] \\
&= \sum_{\mathbf{x}_n \in \mathbf{X}_U} \sum_{k=1}^{K} \mathbb{E}[z_{nk}|\mathbf{x}_n, \boldsymbol{\Theta}^{(t)}](\log \pi_k + \log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)) \\
&\quad + \sum_{\mathbf{x}_n \in \mathbf{X}_L} \sum_{k=1}^{K} \mathbb{E}[z_{nk}|\mathbf{x}_n, y_n, \boldsymbol{\Theta}^{(t)}](\log \pi_k + \log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)) \\
&= \sum_{\mathbf{x}_n \in \mathbf{X}_U} \sum_{k=1}^{K} \gamma(z_{nk})(\log \pi_k + \log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)) \\
&\quad + \sum_{\mathbf{x}_n \in \mathbf{X}_L} \sum_{k=1}^{K} \gamma(z_{nk}|y_n)(\log \pi_k + \log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)). \qquad (4.32)
\end{aligned}$$

The corresponding update rules are then derived as in (4.23, ..., 4.28) and given by

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{\sum_{\mathbf{x}_n \in \mathbf{X}_U} \gamma(z_{nk})\mathbf{x}_n + \sum_{\mathbf{x}_n \in \mathbf{X}_L} \gamma(z_{nk}|y_n)\mathbf{x}_n}{\sum_{\mathbf{x}_n \in \mathbf{X}_U} \gamma(z_{nk}) + \sum_{\mathbf{x}_n \in \mathbf{X}_L} \gamma(z_{nk}|y_n)}, \qquad (4.33)$$

$$\boldsymbol{\Sigma}_k^{(t+1)} = \frac{\sum_{\mathbf{x}_n \in \mathbf{X}_U} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}} + \sum_{\mathbf{x}_n \in \mathbf{X}_L} \gamma(z_{nk}|y_n)(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}}{\sum_{\mathbf{x}_n \in \mathbf{X}_U} \gamma(z_{nk}) + \sum_{\mathbf{x}_n \in \mathbf{X}_L} \gamma(z_{nk}|y_n)}, \qquad (4.34)$$

$$\pi_k^{(t+1)} = \frac{1}{N}\left(\sum_{\mathbf{x}_n \in \mathbf{X}_U} \gamma(z_{nk}) + \sum_{\mathbf{x}_n \in \mathbf{X}_L} \gamma(z_{nk}|y_n)\right). \qquad (4.35)$$

Chapter 5

Temporal model

In the previous chapter our discussion focused on data which we assumed to be independent and identically distributed. However, many real-life datasets exhibit clear temporal patterns with strong correlations between instances. In this chapter we present our temporal mixture model based on Markov random fields (MRFs). We now assume that the states of the latent variables $\mathbf{Z}$ are locally smooth over time. This corresponds to the temporal smoothness assumption from Section 2.2. Intuitively, we expect consecutive latent variables to be predisposed towards the same state.

5.1 Motivation

There are several techniques for dealing with temporal data. One approach is to move a sliding window over the data and combine all instances contained inside into a single feature vector. This way, information about related instances is stored in the new feature vectors directly. However, this approach does not scale well, since the dimensionality of the dataset increases rapidly with the window size. For Gaussian mixture models with full covariance matrices the number of parameters may quickly become prohibitively large [6].

Hidden Markov models (HMMs) are another popular way of modelling temporal data [30]. Instead of treating the latent variables as independent, HMMs assume that each $\mathbf{z}_n$ depends only on the preceding $\mathbf{z}_{n-1}$. Given $\mathbf{z}_{n-1}$, the distribution over the possible states for $\mathbf{z}_n$ is defined in terms of a $K \times K$ transition matrix, where K is again the number of components. The transition matrix specifies the probability to transition from one component to another (or remain unchanged). Figure 5.1 shows a graphical representation of a hidden Markov model.

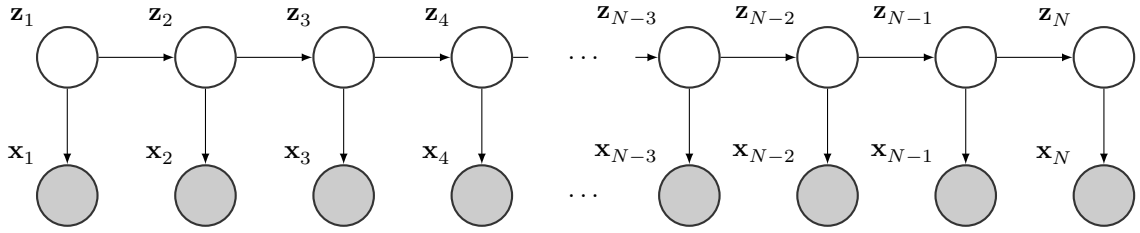


Figure 5.1: A graphical representation of a first-order hidden Markov model.

HMMs have been successfully applied to a variety of problem domains, including speech recognition [17] and bioinformatics [19]. However, they have some shortcomings for our application. First, the assumption that each $\mathbf{z}_n$ depends only on the previous $\mathbf{z}_{n-1}$ is quite restrictive. In bird song detection, for example, the bird songs consist of intricate patterns that typically last multiple seconds. While there exists some research on higher-order HMMs with more complex dependencies, it remains difficult to estimate the parameters of their transition matrices [29, 31].

Second, HMMs mainly excel in modelling sequential data, where each $\mathbf{z}_n$ is only influenced by the past (i.e. the previous instance). This is useful in online environments where we have to make real-time decisions based on the data we have seen so far. However, in the case of rare category detection we already have access to the entire dataset beforehand. As such, we can use information from both preceding and subsequent instances. HMMs are less useful in such cases, as they mainly make inferences in just one direction.

To address these shortcomings we define our temporal mixture model as a Markov random field. This has to distinct advantages compared to HMMs: (1) the graphical models for MRFs are undirected and therefore allow inference from both preceding and subsequent instances, and (2) they are easily scalable to high orders. Traditionally, MRFs are used to model spatial smoothness rather than temporal smoothness. For example, they are used in image segmentation to account for the fact that neighbouring pixels are often similar [39]. We apply MRFs to temporal data by formulating comparable "temporal neighbourhoods", i.e. instances that are close together in time. In the next section we describe MRFs and our proposed model in detail.

5.2 Temporal Markov random fields

MRFs can be represented visually as undirected graphical models. Because the associated graph has no concept of directionality we can formulate relations between "past" and "future" instances. This is not possible in directed graphs because such connections would introduce cycles. When we think of a generative model it may seem counter-intuitive for future instances to affect past instances. However, if we consider the model from a descriptive point of view we can imagine that new information about $\mathbf{z}_n$ also tells us something about the preceding instances $\mathbf{z}_{n-1}, \ldots, \mathbf{z}_{n-h}$.

Here we introduce a new quantity h, which denotes the "neighbourhood radius" and is an essential part of our model definition. It represents the number of nodes in each direction that are directly connected to $\mathbf{z}_n$. For larger h we can model long-range dependencies that extend beyond a single instance. In our example of bird song detection this allows us to capture patterns that last for several seconds. Figure 5.2 shows a graphical representation of our proposed model for $h = 2$. In practice we may prefer $h > 2$ but this would make the visualization unnecessarily cluttered.

Our model as a whole is strictly speaking not an MRF because we only model the latent variables $\mathbf{Z}$ as such. Like before we still assume each $\mathbf{z}_n$ has a Gaussian emission distribution for its corresponding observable $\mathbf{x}_n$ depending on its state. Our model can therefore be more accurately characterized as a hidden Markov random field (HMRF), since the observable variables are not part of the MRF. This approach is comparable to [39], where a spatial HMRF model is proposed for segmentation of brain MR images.

5.2.1 Model definition

Each latent node $\mathbf{z}_n$ is connected to its hth-order neighbours. Each $\mathbf{x}_n$ is still conditionally independent from all other nodes in the graph given its corresponding latent variable $\mathbf{z}_n$. The conditional

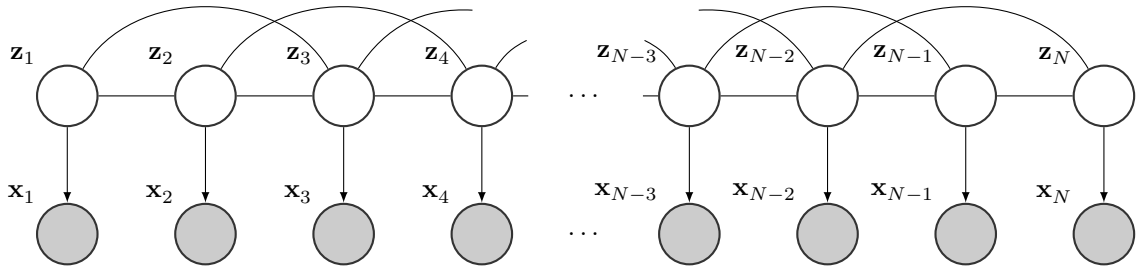


Figure 5.2: A graphical representation of our proposed temporal model. The latent variables $\mathbf{Z}$ are represented by a hidden Markov random field with a neighbourhood radius of $h = 2$. Each $\mathbf{z}_n$ emits a corresponding observable variable $\mathbf{x}_n$ according to a Gaussian emission distribution depending on its state.

distribution of $\mathbf{X}$ given $\mathbf{Z}$ is therefore the same as in the static setting and hence given by

$$p\left(\mathbf{X}|\mathbf{Z}\right) = \prod_{n=1}^{N} p(\mathbf{x}_n|\mathbf{Z}) = \prod_{n=1}^{N} p(\mathbf{x}_n|\mathbf{z}_n). \tag{5.1}$$

The difference only becomes apparent when we consider the marginal distribution $p(\mathbf{Z})$, which can no longer be factorized with respect to n. This is due to the strong interdependence between different $\mathbf{z}_n$. As a result the joint distribution does not factorize over n either and we have

$$p(\mathbf{X}, \mathbf{Z}) = p(\mathbf{Z})p(\mathbf{X}|\mathbf{Z}) = p(\mathbf{Z}) \prod_{n=1}^{N} p(\mathbf{x}_n|\mathbf{z}_n). \tag{5.2}$$

We now seek a different way to factorize $p(\mathbf{Z})$ by considering the conditional independence properties of the undirected subgraph $G_{\mathbf{Z}}$ containing only the nodes for the latent variables $\mathbf{Z}$. Since this graph is undirected two unconnected nodes $\mathbf{z}_i, \mathbf{z}_j$ are conditionally independent given all other nodes $\mathbf{Z}_{\setminus\{i,j\}}$. Formally, for all i, j such that $(\mathbf{z}_i, \mathbf{z}_j) \notin E$ we have

$$p(\mathbf{z}_i, \mathbf{z}_j|\mathbf{Z}_{\setminus\{i,j\}}) = p(\mathbf{z}_i|\mathbf{Z}_{\setminus\{i,j\}})p(\mathbf{z}_j|\mathbf{Z}_{\setminus\{i,j\}}). \tag{5.3}$$

As a result $\mathbf{z}_i$ and $\mathbf{z}_j$ can therefore never appear in the same factor, because if they did this conditional independence property would not hold.

We conclude that the nodes appearing together in each factor must constitute (part of) a clique in the graph. A clique is a subset $C \subseteq V$ of nodes that are fully connected such that $(u, v) \in E$ for all mutually distinct $u, v \in C$. If this would not be the case there would be at least one pair of nodes that are not connected within a single factor, which we have just established is impossible. In our discussion we limit ourselves to maximal cliques for which there exists no node $u \in V \setminus C$ such that $C \cup \{u\}$ would also be a clique. These are sufficient because we can absorb factors of smaller cliques in the factors of their maximal cliques without loss of generality. It is thus possible to express the marginal distribution $p(\mathbf{Z})$ as the product of factors involving the maximal cliques such that

$$p(\mathbf{Z}) = \frac{1}{Z} \prod_{C} \psi_C(\mathbf{Z}_C), \tag{5.4}$$

where the factors $\psi_C(\mathbf{Z}_C)$ are called potential functions and Z is a normalization constant given by

$$Z = \sum_{\mathbf{Z}} \prod_C \psi_C(\mathbf{Z}_C) \tag{5.5}$$

and hence ensures that $\sum_{\mathbf{Z}} p(\mathbf{Z}) = 1$. It is worthwhile to point out that computing the normalization constant Z generally requires an exponential number of computations due to the summation over all possible realizations of $\mathbf{Z}$. If there are N variables that can each have K different states we have to consider K^N terms. This makes it infeasible to compute Z in practice, but we will see later in this section that it is not always necessary to do so.

If $p(\mathbf{Z}) > 0$ for all $\mathbf{Z}$ the Hammersley-Clifford theorem states that such an MRF can be characterized by a Boltzmann distribution [3] so that

$$p(\mathbf{Z}) = \frac{1}{Z} \exp\left(-U(\mathbf{Z})\right), \tag{5.6}$$

where $U(\mathbf{Z})$ is called an energy function. This expression naturally follows from (5.4) if we choose

$$\psi_C(\mathbf{Z}_C) = \exp\left(-F_C(\mathbf{Z}_C)\right), \tag{5.7}$$

which simultaneously ensures that indeed $p(\mathbf{Z}) > 0$. Here, $F_C(\mathbf{Z}_C)$ is the energy associated with a clique C. The total energy of the distribution is then conveniently given by the sum of all clique energies such that

$$U(\mathbf{Z}) = \sum_C F_C(\mathbf{Z}_C). \tag{5.8}$$

Due to the sign in the exponent, clique configurations with low energies correspond to high global probabilities. Since we are free to define the energy function $F_C(\mathbf{Z}_C)$ ourselves we want this to be reflected in our formulation. In order to do so we first have to consider what kind of cliques occur in our model and then decide which configurations are desirable.

In the subgraph $G_{\mathbf{Z}}$ each node $\mathbf{z}_n$ is connected only to the nodes $\{\mathbf{z}_{n-h}, \ldots, \mathbf{z}_{n-1}, \mathbf{z}_{n+1}, \ldots, \mathbf{z}_{n+h}\}$. We call this set of nodes the Markov blanket or neighbourhood of $\mathbf{z}_n$ and denote it by $\mathrm{MB}(\mathbf{z}_n)$. It is clear that any node sharing a clique with $\mathbf{z}_n$ also has to be in $\mathrm{MB}(\mathbf{z}_n)$. For any $\mathbf{z}_i \in \mathrm{MB}(\mathbf{z}_n)$, consider the set of nodes $\{\mathbf{z}_i, \ldots, \mathbf{z}_n\}$ consisting of $\mathbf{z}_i$, $\mathbf{z}_n$ and all nodes in between them. Here we assume $i < n$ but the same claim holds if $i > n$ and we write the set as $\{\mathbf{z}_n, \ldots, \mathbf{z}_i\}$. By design it holds that all these nodes are interconnected and thus form a clique. Recall that it is sufficient to consider only maximal cliques which we obtain by setting $i = n - h$ or $i = n + h$, corresponding to the left and right neighbourhood of $\mathbf{z}_n$, respectively.

Figure 5.3 shows the subgraph $G_{\mathbf{Z}}$ corresponding to a simple hypothetical case with $N = 7$ and $h = 2$. Observe that any node $\mathbf{z}_n$ with $n \leq h$ or $n > N - h$ has cliques to their left or right that are subsets of maximal cliques corresponding to the left or right neighbourhoods of $\mathbf{z}_{h+1}$ or $\mathbf{z}_{N-h}$, respectively. We also need to account for duplicate maximal cliques. For example, the clique corresponding to the right neighbourhood of $\mathbf{z}_3$ is identical to the clique corresponding to the left neighbourhood of $\mathbf{z}_5$. In general we see that there are $N-h$ unique maximal cliques in any subgraph $G_{\mathbf{Z}}$, each of size $h+1$. We denote these cliques by C_i and the corresponding set of latent variables by $\mathbf{Z}_{C_i} = \{\mathbf{z}_i, \ldots, \mathbf{z}_{i+h}\}$ with $1 \leq i \leq N - h$.

Now that the different possible maximal cliques are clearly defined we can discuss which configurations are desirable. Recall the motivation for modelling the temporal aspect of our data: the assumption that class labels are temporally smooth. From this perspective it makes sense if the

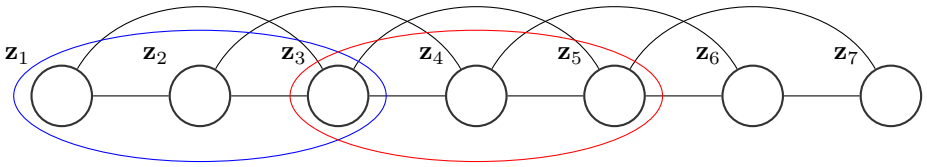


Figure 5.3: A simplified visualization of the subgraph $G_{\mathbf{Z}}$ with $N = 7$ and $h = 2$. Two maximal cliques corresponding to the left and right neighbourhood of $\mathbf{z}_3$ are outlined in blue and red, respectively.

energy associated with a clique is a measure of the local smoothness. We therefore define our clique energy function as

$$F(\mathbf{Z}_C) = -\sum_{\{i,j|j>i\}\subseteq C} \mathbf{z}_i^{\mathrm{T}} \mathbf{z}_j, \tag{5.9}$$

where we are taking the negative sum of inner products of all mutually distinct pairs $\mathbf{z}_i$ and $\mathbf{z}_j$. Recall that both vectors have a 1-of-K representation so that $\mathbf{z}_i^{\mathrm{T}} \mathbf{z}_j = 1$ if they are equal and 0 otherwise. We can therefore quickly verify that this function has the desired property of assigning low energies to homogeneous cliques.

5.2.2 Toy example

Let us now consider an example to illustrate the properties of this energy function. For this we again turn to Figure 5.3. The unique maximal cliques are given by

$$\begin{aligned}
\mathbf{Z}_{C_1} &= \{\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3\}, \\
\mathbf{Z}_{C_2} &= \{\mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4\}, \\
\mathbf{Z}_{C_3} &= \{\mathbf{z}_3, \mathbf{z}_4, \mathbf{z}_5\}, \\
\mathbf{Z}_{C_4} &= \{\mathbf{z}_4, \mathbf{z}_5, \mathbf{z}_6\}, \\
\mathbf{Z}_{C_5} &= \{\mathbf{z}_5, \mathbf{z}_6, \mathbf{z}_7\}.
\end{aligned}$$

Given any configuration of the state matrix $\mathbf{Z}$ we can now compute the clique energies. For now we assume there are only $K = 2$ possible states for each latent variable, though the extension to more states is straightforward. We define a state sequence $k_1 \dots k_N$ as a shorthand to denote the state of each variable $\mathbf{z}_n$ so that $z_{nk_n} = 1$. Consider for example the state sequence 1222112 which would correspond to the state matrix

$$\mathbf{Z} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{5.10}$$

The corresponding clique energies can then be computed directly. For clarity and completeness we show the computations and resulting energies below, though they are quite elementary.

$$F(\mathbf{Z}_{C_1}) = -\mathbf{z}_1^{\mathrm{T}}\mathbf{z}_2 - \mathbf{z}_1^{\mathrm{T}}\mathbf{z}_3 - \mathbf{z}_2^{\mathrm{T}}\mathbf{z}_3 = -1 \tag{5.11}$$

$$F(\mathbf{Z}_{C_2}) = -\mathbf{z}_2^{\mathrm{T}}\mathbf{z}_3 - \mathbf{z}_2^{\mathrm{T}}\mathbf{z}_4 - \mathbf{z}_3^{\mathrm{T}}\mathbf{z}_4 = -3 \tag{5.12}$$

$$F(\mathbf{Z}_{C_3}) = -\mathbf{z}_3^{\mathrm{T}}\mathbf{z}_4 - \mathbf{z}_3^{\mathrm{T}}\mathbf{z}_5 - \mathbf{z}_4^{\mathrm{T}}\mathbf{z}_5 = -1 \tag{5.13}$$

$$F(\mathbf{Z}_{C_4}) = -\mathbf{z}_4^{\mathrm{T}}\mathbf{z}_5 - \mathbf{z}_4^{\mathrm{T}}\mathbf{z}_6 - \mathbf{z}_5^{\mathrm{T}}\mathbf{z}_6 = -1 \tag{5.14}$$

$$F(\mathbf{Z}_{C_5}) = -\mathbf{z}_5^{\mathrm{T}}\mathbf{z}_6 - \mathbf{z}_5^{\mathrm{T}}\mathbf{z}_7 - \mathbf{z}_6^{\mathrm{T}}\mathbf{z}_7 = -1 \tag{5.15}$$

The second cluster stands out as having the lowest energy and should therefore have the most desirable configuration. Indeed we see that all three vectors in $\mathbf{Z}_{C_2}$ are identical indicating that this clique is perfectly smooth. The total energy is simply the sum of all clique energies so we have $U(\mathbf{Z}) = -7$ and consequently $p(\mathbf{Z}) = \frac{1}{Z}e^7$.

5.2.3 Neighbourhood-conditional probabilities

As we stated before it is computationally infeasible to determine the value of Z since it requires summing over all K^N possible state matrices $\mathbf{Z}$. However, if we are only interested in conditional distributions we do not actually need to know Z. Consider for example the conditional distribution $p(\mathbf{z}_n|\mathbf{Z}_{\setminus\{n\}})$ of a single node $\mathbf{z}_n$ given all other nodes. Using Bayes' theorem we can write

$$\begin{aligned} p(\mathbf{z}_n|\mathbf{Z}_{\setminus\{n\}}) &= \frac{p(\mathbf{Z})}{\sum_{k=1}^{K} p(\mathbf{Z}_{nk\leftarrow 1})} \\ &= \frac{Z^{-1}\exp\left(-U(\mathbf{Z})\right)}{\sum_{k=1}^{K} Z^{-1}\exp\left(-U\left(\mathbf{Z}_{nk\leftarrow 1}\right)\right)} \\ &= \frac{\exp\left(-U(\mathbf{Z})\right)}{\sum_{k=1}^{K}\exp\left(-U\left(\mathbf{Z}_{nk\leftarrow 1}\right)\right)} \end{aligned} \tag{5.16}$$

where we use $\mathbf{Z}_{nk\leftarrow 1}$ to denote the same set as $\mathbf{Z}$ but with $\mathbf{z}_n$ changed so that its kth entry now equals 1 (and consequently the entry that previously equalled 1 is set to 0). Notice how the terms involving Z cancel out between the numerator and denominator so that we are no longer required to explicitly compute Z.

In MRFs only neighbouring nodes directly affect each other, so the distribution of a single variable conditioned only on its neighbours is unaffected when we condition on any additional nodes. Formally, this property is expressed as

$$p(\mathbf{z}_n|\mathbf{Z}_{\setminus\{n\}}) = p(\mathbf{z}_n|\mathrm{MB}(\mathbf{z}_n)). \tag{5.17}$$

So instead of considering the distribution of $\mathbf{z}_n$ given all other nodes it is sufficient to condition only on $\mathrm{MB}(\mathbf{z}_n)$. We call $p(\mathbf{z}_n|\mathrm{MB}(\mathbf{z}_n))$ the neighbourhood-conditional probability of $\mathbf{z}_n$. This makes sense intuitively because in (5.16) all clique energies for cliques not involving $\mathbf{z}_n$ are shared between

the numerator and denominator, so they also cancel out. This leaves only the cliques $C_{n-h}, \ldots, C_n$ which contain no other nodes than those in $\mathrm{MB}(\mathbf{z}_n)$. Combining (5.16) and (5.17) we obtain

$$\begin{aligned} p(\mathbf{z}_n|\mathrm{MB}(\mathbf{z}_n)) &= \frac{\exp\left(-U\left(\mathrm{MB}(\mathbf{z}_n) \cup \{\mathbf{z}_n\}\right)\right)}{\sum_{k=1}^{K} \exp\left(-U\left(\mathrm{MB}(\mathbf{z}_n) \cup \{\mathbf{z}_{nk\leftarrow 1}\}\right)\right)} \\ &= \frac{\exp\left(-\sum_{i=0}^{h} F\left(\mathbf{Z}_{C_{n-i}}\right)\right)}{\sum_{k=1}^{K} \exp\left(-\sum_{i=0}^{h} F\left(\mathbf{Z}_{C_{n-i},nk\leftarrow 1}\right)\right)}, \end{aligned} \tag{5.18}$$

where $\mathbf{z}_{nk\leftarrow 1}$ and $\mathbf{Z}_{C_{n-i},nk\leftarrow 1}$ are defined analogously to $\mathbf{Z}_{nk\leftarrow 1}$ and denote that the value of $\mathbf{z}_n$ is changed so that now $z_{nk} = 1$.

With our choice of energy function the neighbourhood-conditional distributions implicitly assign higher weights to neighbours that are close to $\mathbf{z}_n$ relative to more distant neighbours. We can see this by considering $p(\mathbf{z}_3|\mathrm{MB}(\mathbf{z}_3))$ for the state matrix given in (5.10). Looking at just the states of the neighbours $\mathrm{MB}(\mathbf{z}_3) = \{\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_4, \mathbf{z}_5\}$ they are evenly split between state 1 and 2, so we might expect to find

$$p(z_{31} = 1|\mathrm{MB}(\mathbf{z}_3)) = p(z_{32} = 1|\mathrm{MB}(\mathbf{z}_3)) = 0.5. \tag{5.19}$$

However, if we use (5.18) we actually find a different result. We first compute $U\left(\mathrm{MB}(\mathbf{z}_3) \cup \{\mathbf{z}_{3k\leftarrow 1}\}\right)$ for each value of k. For $k = 1$ the corresponding state sequence is given by 1212112, whereas for $k = 2$ we can use the same state sequence for the state matrix in (5.10) since $z_{32} = 1$ already. Using the same strategy as in (5.11, …, 5.15) we find

$$U\left(\mathrm{MB}(\mathbf{z}_3) \cup \{\mathbf{z}_{31\leftarrow 1}\}\right) = -3, \tag{5.20}$$

$$U\left(\mathrm{MB}(\mathbf{z}_3) \cup \{\mathbf{z}_{32\leftarrow 1}\}\right) = -5. \tag{5.21}$$

The proper neighbourhood-conditional probabilities are then given by

$$p(z_{31} = 1|\mathrm{MB}(\mathbf{z}_3)) = \frac{e^3}{e^3 + e^5} = 0.12, \tag{5.22}$$

$$p(z_{32} = 1|\mathrm{MB}(\mathbf{z}_3)) = \frac{e^5}{e^3 + e^5} = 0.88, \tag{5.23}$$

which are decidedly in favour of state 2. This makes sense once we also take the ordering of the neighbours into account. Both $\mathbf{z}_2$ and $\mathbf{z}_4$ are in state 2 and also closest to $\mathbf{z}_3$, while the neighbours $\mathbf{z}_1$ and $\mathbf{z}_5$ in state 1 are further away.

The reason this happens, even though we did not explicitly encode this behaviour into our energy function, is because close neighbours appear in more terms of the summation in (5.8). To illustrate this, Table 5.1 shows an overview of all inner products involved in the computation of $p(\mathbf{z}_3|\mathrm{MB}(\mathbf{z}_3))$. Importantly, the same term may appear in multiple cliques. A tally of the different terms involving $\mathbf{z}_3$ shows that $\mathbf{z}_1^{\mathrm{T}}\mathbf{z}_3$ and $\mathbf{z}_3^{\mathrm{T}}\mathbf{z}_5$ appear only once, whereas $\mathbf{z}_2^{\mathrm{T}}\mathbf{z}_3$ and $\mathbf{z}_3^{\mathrm{T}}\mathbf{z}_4$ both appear twice. Due to symmetry we have that for any $p(\mathbf{z}_n|\mathrm{MB}(\mathbf{z}_n))$ the terms $\mathbf{z}_n^{\mathrm{T}}\mathbf{z}_{n-h}$ and $\mathbf{z}_n^{\mathrm{T}}\mathbf{z}_{n+h}$ appear once, $\mathbf{z}_n^{\mathrm{T}}\mathbf{z}_{n-h+1}$ and $\mathbf{z}_n^{\mathrm{T}}\mathbf{z}_{n+h-1}$ appear twice, and so on.

The other interesting thing to note about Table 5.1 is that there are still some inner products that do not involve $\mathbf{z}_3$ at all. All such inner products are again shared between the numerator and

Table 5.1: An overview of the different inner products involved in computing $p(\mathbf{z}_3|\mathrm{MB}(\mathbf{z}_3))$.

Clique	Inner products		
C_1	$\mathbf{z}_1^\mathrm{T}\mathbf{z}_2$	$\mathbf{z}_1^\mathrm{T}\mathbf{z}_3$	$\mathbf{z}_2^\mathrm{T}\mathbf{z}_3$
C_2	$\mathbf{z}_2^\mathrm{T}\mathbf{z}_3$	$\mathbf{z}_2^\mathrm{T}\mathbf{z}_4$	$\mathbf{z}_3^\mathrm{T}\mathbf{z}_4$
C_3	$\mathbf{z}_3^\mathrm{T}\mathbf{z}_4$	$\mathbf{z}_3^\mathrm{T}\mathbf{z}_5$	$\mathbf{z}_4^\mathrm{T}\mathbf{z}_5$

denominator of (5.18) and therefore cancel out. By ignoring all inner products not involving $\mathbf{z}_n$ and properly weighing those that do we can also write (5.18) as

$$\begin{aligned} p(\mathbf{z}_n|\mathrm{MB}(\mathbf{z}_n)) &= \frac{\exp\left(\sum\limits_{\mathbf{z}_i\in\mathrm{MB}(\mathbf{z}_n)} w_i \mathbf{z}_i^\mathrm{T}\mathbf{z}_n\right)}{\sum\limits_{k=1}^{K}\exp\left(\sum\limits_{\mathbf{z}_i\in\mathrm{MB}(\mathbf{z}_n)} w_i \mathbf{z}_i^\mathrm{T}\mathbf{z}_{nk\leftarrow 1}\right)} \\ &\propto \exp\left(\sum_{\mathbf{z}_i\in\mathrm{MB}(\mathbf{z}_n)} w_i \mathbf{z}_i^\mathrm{T}\mathbf{z}_n\right), \end{aligned} \tag{5.24}$$

where w_i denotes the number of times the inner product $\mathbf{z}_i^\mathrm{T}\mathbf{z}_n$ appears and is given by

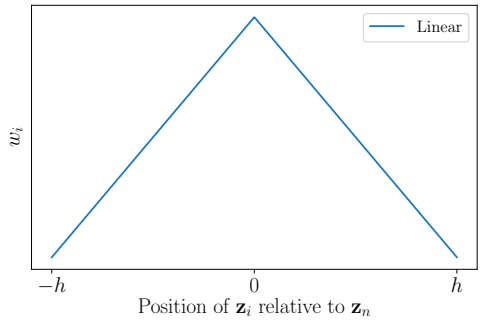
$$w_i = \begin{cases} h+1-|i-n| & \text{if } \mathbf{z}_i \in \mathrm{MB}(\mathbf{z}_n) \\ 0 & \text{otherwise.} \end{cases} \tag{5.25}$$

In the remainder we refer to w_i as the weight of $\mathbf{z}_i$. Note that the concept of cliques does not enter into (5.24) anymore, so we no longer have to reason about which cliques or nodes are involved in computing $p(\mathbf{z}_n|\mathrm{MB}(\mathbf{z}_n))$. Instead we only have to compute the inner products $\mathbf{z}_n^\mathrm{T}\mathbf{z}_{n-h}, \ldots, \mathbf{z}_n^\mathrm{T}\mathbf{z}_{n-1}, \mathbf{z}_n^\mathrm{T}\mathbf{z}_{n+1}, \ldots, \mathbf{z}_n^\mathrm{T}\mathbf{z}_{n+h}$ and weigh them according to (5.25).

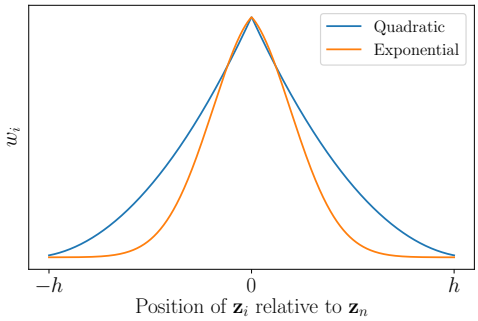
For the energy function (5.9) we have been using until now the weights are given by a piecewise linear function of the relative node positions. Figure 5.4a shows a graphical representation of its shape. This linearity is rather restrictive, so it is natural to wonder whether we can also obtain different shapes. It turns out that by updating our energy function we can do so indeed. In particular, multiplicative terms applied to the inner product inside the summation of (5.9) are preserved so that they enter directly into the summation of (5.24). They can therefore be absorbed into the weights w_i if we also update (5.25) accordingly.

We are especially interested in multiplicative terms $f_h(d)$ that are functions of the distance $d = |i-j|$ between two nodes $\mathbf{z}_i$ and $\mathbf{z}_j$. Such terms have the desirable property that the resulting energy function remains symmetric and invariant to translation. We can then write the updated energy function as

$$F(\mathbf{Z}_C) = -\sum_{\{i,j|j>i\}\subseteq C} f_h(|i-j|)\mathbf{z}_i^\mathrm{T}\mathbf{z}_j. \tag{5.26}$$



(a) Shape of the weight function corresponding to (5.9).



(b) Shapes of the weight functions corresponding to (5.28) and (5.29).

Figure 5.4: The shapes of various weight functions corresponding to different expressions for the clique energies.

The new weights are then given by

$$w_i = \begin{cases} f_h(|i-n|)(h+1-|i-n|) & \text{if } \mathbf{z}_i \in \text{MB}(\mathbf{z}_n) \\ 0 & \text{otherwise.} \end{cases} \tag{5.27}$$

Note that in (5.26) $f_h(|i-j|)$ is evaluated for all $\{i, j | j > i\} \subseteq C$, whereas in (5.27) it is sufficient to consider $f_h(|i-n|)$ for fixed n. This is because in (5.26) we measure the energy of an entire clique in which no single variable can be considered special. Meanwhile, (5.27) is involved only in the computation of $p(\mathbf{z}_n|\text{MB}(\mathbf{z}_n))$ where we have seen that $\mathbf{z}_n$ does hold special significance.

Figure 5.4b shows the shapes of two different weight distributions, which we normalize so their shapes can be easily compared on a similar scale. Specifically, we plot the weight distributions corresponding to

$$f_h(d) = h + 1 - d \tag{5.28}$$

$$f_h(d) = \exp\left(-\frac{8d^2}{h^2}\right), \tag{5.29}$$

where (5.28) results in a piecewise quadratic function and (5.29) causes the weights to be approximately normally distributed with a standard deviation of $\sigma = \frac{h}{4}$.

5.2.4 Neighbourhood-conditional responsibilities

So far we have mainly been concerned with finding an expression for the neighbourhood-conditional probabilities $p(\mathbf{z}_n|\text{MB}(\mathbf{z}_n))$. The reason we give such a detailed treatment of this quantity is because it plays a crucial role in deriving new update rules for our temporal model. For this we would again like to use the EM algorithm. However, since each latent variable now also depends on the state of its neighbourhood, we can no longer use the same approach as in Section 4.2. To see why, recall that we previously assumed that each $\mathbf{z}_n$ came from a categorical distribution such that we simply

had $p(z_{nk} = 1) = \pi_k$. We then used this to obtain a tractable expression for the responsibilities $\gamma(z_{nk})$ given by (4.20).

In our temporal model it has become infeasible to compute the responsibilities the same way because we have

$$p(z_{nk} = 1) = \sum_{\mathrm{MB}(\mathbf{z}_n)} p(\mathrm{MB}(\mathbf{z}_n))p(z_{nk} = 1|\mathrm{MB}(\mathbf{z}_n)), \tag{5.30}$$

where the summation is over all possible configurations of the neighbourhood $\mathrm{MB}(\mathbf{z}_n)$. Since there are $\mathcal{O}(h)$ nodes in $\mathrm{MB}(\mathbf{z}_n)$ and each node has K possible states, there are $\mathcal{O}(h^K)$ unique configurations to consider. It is also unclear how to compute $p(\mathrm{MB}(\mathbf{z}_n))$ in the first place. On the other hand, if we know the state of the neighbourhood beforehand, computing $p(z_{nk} = 1|\mathrm{MB}(\mathbf{z}_n))$ is easy. Our strategy is therefore as follows: in the E-step we first find an estimate $\hat{\mathbf{Z}}$ of the true configuration given $\mathbf{X}$ and $\boldsymbol{\Theta}^{(t)}$, and then condition on $\hat{\mathbf{Z}}$ so we only have to compute the neighbourhood-conditional probabilities.

Leaving out the dependency on the parameters $\boldsymbol{\Theta}^{(t)}$ for clarity, the maximum a posteriori (MAP) estimate $\hat{\mathbf{Z}}$ is given by

$$\hat{\mathbf{Z}} = \arg\max_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}) = \arg\max_{\mathbf{Z}} p(\mathbf{Z})p(\mathbf{X}|\mathbf{Z}). \tag{5.31}$$

This is different from the maximum likelihood estimate in that it also accounts for the prior probability $p(\mathbf{Z})$. Note that this maximization is actually also intractable because it requires consideration of all N^K possible configurations of $\mathbf{Z}$. However, we are at least able to evaluate the objective function for any given $\mathbf{Z}$ using (5.1) and (5.4). We would normally be unable to use (5.4) directly due to the normalization constant, but since we are only interested in maximizing (5.31) we can ignore it. This suggests we may be able to approximate the solution using a greedy algorithm.

A commonly used iterative approximation algorithm is the iterated conditional modes (ICM) algorithm proposed in [2]. Given the observed data $\mathbf{X}$ and a current estimate $\hat{\mathbf{Z}}$, ICM updates its estimate of each individual $\hat{\mathbf{z}}_n$ in turn by maximizing the posterior probability $p(\mathbf{z}_n|\mathbf{X}, \hat{\mathbf{Z}}_{\setminus\{n\}})$ with respect to $\mathbf{z}_n$. Because $p(\hat{\mathbf{Z}}|\mathbf{X}) = p(\hat{\mathbf{Z}}_{\setminus\{n\}}|\mathbf{X})p(\hat{\mathbf{z}}_n|\mathbf{X}, \hat{\mathbf{Z}}_{\setminus\{n\}})$ and the algorithm always maximizes $p(\hat{\mathbf{z}}_n|\mathbf{X}, \hat{\mathbf{Z}}_{\setminus\{n\}})$, it is guaranteed that $p(\hat{\mathbf{Z}}|\mathbf{X})$ never decreases. One cycle in which each $\hat{\mathbf{z}}_n$ is considered, constitutes a single iteration of the algorithm which is repeated until convergence (or until a predefined number of iterations have been completed). Importantly, each iteration only requires $\mathcal{O}(NK)$ computations so the algorithm runs in polynomial time.

We have not previously considered the posterior probability $p(\mathbf{z}_n|\mathbf{X}, \hat{\mathbf{Z}}_{\setminus\{n\}})$ so it is useful to give it some more thought. We first observe that conditioning on just $\mathbf{x}_n$ instead of $\mathbf{X}$ is sufficient because $\mathbf{z}_n$ and $\mathbf{X}_{\setminus\{n\}}$ are conditionally independent given $\hat{\mathbf{Z}}_{\setminus\{n\}}$. We thus have $p(\mathbf{z}_n|\mathbf{X}, \hat{\mathbf{Z}}_{\setminus\{n\}}) = p(\mathbf{z}_n|\mathbf{x}_n, \hat{\mathbf{Z}}_{\setminus\{n\}})$. From Bayes' theorem we know that

$$\begin{aligned} p(\mathbf{z}_n|\mathbf{x}_n, \hat{\mathbf{Z}}_{\setminus\{n\}}) &= \frac{p(\mathbf{z}_n|\hat{\mathbf{Z}}_{\setminus\{n\}})p(\mathbf{x}_n|\mathbf{z}_n, \hat{\mathbf{Z}}_{\setminus\{n\}})}{p(\mathbf{x}_n|\hat{\mathbf{Z}}_{\setminus\{n\}})} \\ &\propto p(\mathbf{z}_n|\hat{\mathbf{Z}}_{\setminus\{n\}})p(\mathbf{x}_n|\mathbf{z}_n, \hat{\mathbf{Z}}_{\setminus\{n\}}). \end{aligned} \tag{5.32}$$

Also, recall that $\mathbf{x}_n$ and $\hat{\mathbf{Z}}_{\setminus\{n\}}$ are conditionally independent given $\mathbf{z}_n$ so that $p(\mathbf{x}_n|\mathbf{z}_n, \hat{\mathbf{Z}}_{\setminus\{n\}}) = p(\mathbf{x}_n|\mathbf{z}_n)$. Combining this with (5.17) we obtain

$$\arg\max_{\mathbf{z}_n} p(\mathbf{z}_n|\mathbf{x}_n, \hat{\mathbf{Z}}_{\setminus\{n\}}) = \arg\max_{\mathbf{z}_n} p(\mathbf{z}_n|\mathrm{MB}(\hat{\mathbf{z}}_n))p(\mathbf{x}_n|\mathbf{z}_n), \tag{5.33}$$

meaning we only have to compute the neighbourhood-conditional probability of $\mathbf{z}_n$ using (5.24) and the likelihood of $\mathbf{x}_n$ given by (4.9). Note that both $p(\mathbf{z}_n|\mathrm{MB}(\mathbf{z}_n))$ and $p(\mathbf{x}_n|\mathbf{z}_n)$ belong to the exponential family, so we could simplify this further by maximizing the logarithm instead.

We can now update the expression for $Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(t)})$ for our temporal model as follows:

$$\begin{aligned}
Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(t)}) &= \mathbb{E}_{\mathbf{Z}}\left[\log L_C(\boldsymbol{\Theta}) \;\middle|\; \mathbf{X}, \hat{\mathbf{Z}}, \boldsymbol{\Theta}^{(t)}\right] \\
&= \mathbb{E}_{\mathbf{Z}}\left[\log p(\mathbf{Z}) + \log p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\Theta}) \;\middle|\; \mathbf{X}, \hat{\mathbf{Z}}, \boldsymbol{\Theta}^{(t)}\right] \\
&= \mathbb{E}_{\mathbf{Z}}\left[\log p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\Theta}) \;\middle|\; \mathbf{X}, \hat{\mathbf{Z}}, \boldsymbol{\Theta}^{(t)}\right] + \text{const} \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{E}[z_{nk}|\mathbf{X}, \hat{\mathbf{Z}}, \boldsymbol{\Theta}^{(t)}] \log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \text{const} \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} p(z_{nk} = 1|\mathbf{x}_n, \mathrm{MB}(\hat{\mathbf{z}}_n), \boldsymbol{\Theta}^{(t)}) \log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \text{const.}
\end{aligned} \tag{5.34}$$

The most important thing to note is that we now also condition on $\hat{\mathbf{Z}}$. Since $Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(t)})$ is a function of $\boldsymbol{\Theta}$ and $\boldsymbol{\Theta}^{(t)}$ alone we can treat $\log p(\mathbf{Z})$ as a constant. We also see that by computing $\hat{\mathbf{Z}}$ at the start of the E-step we indeed prevent having to compute the summation in (5.30). Instead, we only have to consider $p(z_{nk} = 1|\mathbf{x}_n, \mathrm{MB}(\hat{\mathbf{z}}_n), \boldsymbol{\Theta}^{(t)})$, which looks similar to the responsibilities $\gamma(z_{nk})$. Here, too, the difference is that we now condition on $\mathrm{MB}(\hat{\mathbf{z}}_n)$ and so we refer to them as the neighbourhood-conditional responsibilities. We denote them by $\xi(z_{nk})$ to clearly distinguish them from $\gamma(z_{nk})$ and obtain

$$\begin{aligned}
\xi(z_{nk}) \equiv p(z_{nk} = 1|\mathbf{x}_n, \mathrm{MB}(\hat{\mathbf{z}}_n)) &= \frac{p(z_{nk} = 1|\mathrm{MB}(\hat{\mathbf{z}}_n))p(\mathbf{x}_n|z_{nk} = 1)}{p(\mathbf{x}_n|\mathrm{MB}(\hat{\mathbf{z}}_n))} \\
&= \frac{p(z_{nk} = 1|\mathrm{MB}(\hat{\mathbf{z}}_n))p(\mathbf{x}_n|z_{nk} = 1)}{\sum_{j=1}^{K} p(z_{nj} = 1|\mathrm{MB}(\hat{\mathbf{z}}_n))p(\mathbf{x}_n|z_{nj} = 1)},
\end{aligned} \tag{5.35}$$

where we have again left out the dependence on the parameters $\boldsymbol{\Theta}^{(t)}$ for clarity.

It is important to emphasize that we no longer recognize $\boldsymbol{\pi}$ as a parameter because the distribution of latent variables is now governed by (5.4). In the M-step we therefore only update $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. If we write out (5.34) as

$$Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(t)}) = \sum_{n=1}^{N}\sum_{k=1}^{K} \xi(z_{nk})\Big(-\frac{1}{2}\log|\boldsymbol{\Sigma}_k| - \frac{1}{2}(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_k)\Big) + \text{const}, \tag{5.36}$$

we see that its functional form is identical to (4.22), except the term $\log \pi_k$ no longer appears and

$\gamma(z_{nk})$ is replaced with $\xi(z_{nk})$. This means the new update rules for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are easily given by

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{\sum_{n=1}^{N} \xi(z_{nk}) \mathbf{x}_n}{\sum_{n=1}^{N} \xi(z_{nk})}, \tag{5.37}$$

$$\boldsymbol{\Sigma}_k^{(t+1)} = \frac{\sum_{n=1}^{N} \xi(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}}{\sum_{n=1}^{N} \xi(z_{nk})}. \tag{5.38}$$

The extension to the semi-supervised setting is also straightforward since we can apply the same logic to (4.32). We then obtain

$$\begin{aligned}
Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(t)}) &= \mathbb{E}_{\mathbf{Z}} \left[\log L_C(\boldsymbol{\Theta}) \;\middle|\; \mathbf{X}_U, \mathbf{X}_L, \mathbf{Y}_L, \hat{\mathbf{Z}}, \boldsymbol{\Theta}^{(t)} \right] \\
&= \sum_{\mathbf{x}_n \in \mathbf{X}_U} \sum_{k=1}^{K} \mathbb{E}[z_{nk} | \mathbf{x}_n, \hat{\mathbf{Z}}, \boldsymbol{\Theta}^{(t)}] \log \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \\
&\quad + \sum_{\mathbf{x}_n \in \mathbf{X}_L} \sum_{k=1}^{K} \mathbb{E}[z_{nk} | \mathbf{x}_n, y_n, \hat{\mathbf{Z}}, \boldsymbol{\Theta}^{(t)}] \log \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \text{const} \\
&= \sum_{\mathbf{x}_n \in \mathbf{X}_U} \sum_{k=1}^{K} \xi(z_{nk}) \log \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \\
&\quad + \sum_{\mathbf{x}_n \in \mathbf{X}_L} \sum_{k=1}^{K} \xi(z_{nk} | y_n) \log \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \text{const},
\end{aligned} \tag{5.39}$$

where $\xi(z_{nk}|y_n)$ is defined analogously to $\gamma(z_{nk}|y_n)$ in (4.30) and thus given by

$$\xi(z_{nk}|y_n) = \begin{cases} \frac{\xi(z_{nk})}{\sum_{j \mapsto y_n} \xi(z_{nj})} & \text{if } k \mapsto y_n \\ 0 & \text{otherwise.} \end{cases} \tag{5.40}$$

It is worth noting that in the estimation of $\hat{\mathbf{Z}}$ we now have to ignore configurations that would be prohibited by the known class labels $\mathbf{Y}_L$. If we know a given y_n we only allow configurations where $\hat{z}_{nk} = 1$ if $k \mapsto y_n$. For the sake of completeness we also give the semi-supervised update rules for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ below:

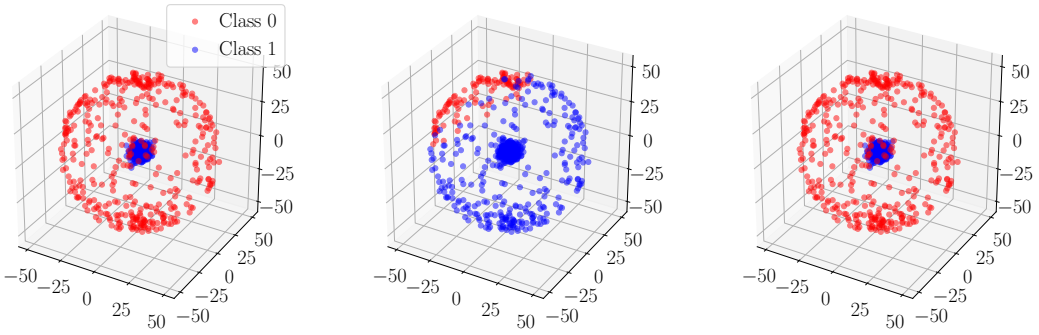
$$\boldsymbol{\mu}_k^{(t+1)} = \frac{\sum_{\mathbf{x}_n \in \mathbf{X}_U} \xi(z_{nk}) \mathbf{x}_n + \sum_{\mathbf{x}_n \in \mathbf{X}_L} \xi(z_{nk}|y_n) \mathbf{x}_n}{\sum_{\mathbf{x}_n \in \mathbf{X}_U} \xi(z_{nk}) + \sum_{\mathbf{x}_n \in \mathbf{X}_L} \xi(z_{nk}|y_n)}, \tag{5.41}$$

$$\boldsymbol{\Sigma}_k^{(t+1)} = \frac{\sum_{\mathbf{x}_n \in \mathbf{X}_U} \xi(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}} + \sum_{\mathbf{x}_n \in \mathbf{X}_L} \xi(z_{nk}|y_n)(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}}{\sum_{\mathbf{x}_n \in \mathbf{X}_U} \xi(z_{nk}) + \sum_{\mathbf{x}_n \in \mathbf{X}_L} \xi(z_{nk}|y_n)}. \tag{5.42}$$

5.3 Comparison with static models

Before we discuss our proposed rare category detection algorithm, it is helpful to first understand the advantages of our temporal HMRF model versus the static mixture model we discuss in Chapter 4. The fundamental difference is that our temporal model effectively replaces the mixture weights $\pi_k = p(z_{nk} = 1)$ with neighbourhood-conditional probabilities $p(z_{nk} = 1|\text{MB}(\mathbf{z}_n))$ that vary per instance. This allows our model to incorporate temporal context to better estimate the component memberships for each instance (or, equivalently, the responsibilities of the components).

To see the effects of this adaptation we visualize the results of both models on a number of synthetic datasets from the Fundamental Clustering Problem Suite (FCPS) [35]. The instances in these datasets are typically assumed to be independent and identically distributed. To simulate the temporal smoothness assumption we treat the data as a time series and group instances from the same class together "in time". In each of the following examples we initialize the components using K-Means clustering. For the temporal model we use a neighbourhood radius of $h = 2$.

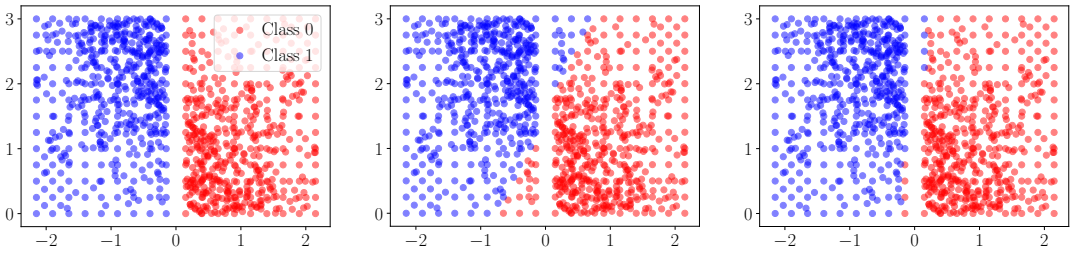


(a) Ground truth (b) Static model (c) Temporal model

Figure 5.5: The Atom dataset.

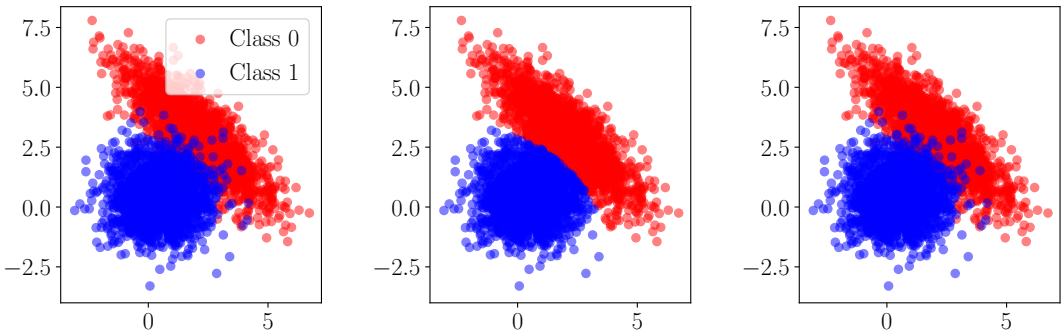
The first synthetic dataset, Atom, is shown in Figure 5.5. The true class labels are shown in Figure 5.5a, while Figure 5.5b and 5.5c show the predicted component memberships for the static and temporal model, respectively. The data is best described as two ball-like clusters that are contained within each other. Because of the shape of the data any Gaussian mixture model is unable to yield an accurate generative model. However, as demonstrated by our temporal model, accurate discriminative models are still possible. By making use of the contextual information our temporal model correctly concludes that instances from the same "ball" likely belong to the same class. This information is then used to improve the model parameters. Meanwhile, the static model is incapable of learning this structure from the data.

Figure 5.6 shows the Wingnut dataset, which consists of two rectangular classes with regions of varying densities. The difficulty is that the low-density region of one class is very close to the high-density region of the other. As a result some of the low-density instances are often misclassified. Indeed, this happens for both the static and temporal model. Note however that the number of misclassified instances is smaller in our temporal model. By making use of temporal information the parameters learned by our model lead to better discriminative performance.



(a) Ground truth (b) Static model (c) Temporal model

Figure 5.6: The Wingnut dataset.



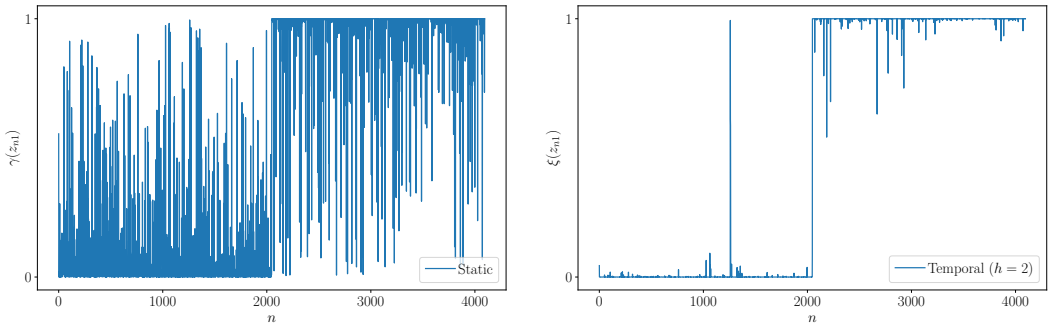
(a) Ground truth (b) Static model (c) Temporal model

Figure 5.7: The Engytime dataset.

The final dataset, Engytime, is shown in Figure 5.7. It consists of two partially overlapping Gaussian classes and should therefore be perfectly suited for both models. Indeed, both the static and temporal model are able to learn the shapes of the two classes. The difference between them becomes apparent when we consider the class membership predictions of each model. The static model is unable to deal with the overlap between the classes, shown by the fact that the few blue instances overlapping with the red class are misclassified. Our temporal model predicts the class membership for almost all instances correctly despite some of them lying deep within the other class.

This dataset also nicely illustrates the effects of the neighbourhood radius h on the neighbourhood-conditional responsibilities computed by our temporal model. Figure 5.8 shows the static responsibilities $\gamma(z_{n1})$ and the neighbourhood-conditional responsibilities $\xi(z_{n1})$ for $h = 2$. It is sufficient to plot the responsibility of the first component, since we only use two components and their responsibilities always sum to 1. The static responsibilities are noisy, while our temporal model clearly shows the temporal smoothness of the classes.

An overview of the performance of both models on each of the previously discussed datasets



(a) Static model

(b) Temporal model

Figure 5.8: The responsibilities for all instances in the Engytime dataset as computed by a static and temporal model, respectively. We simulate temporal smoothness by making sure the first 2048 correspond to one class and the remaining 2048 correspond to the other.

is shown in Table 5.2. An important thing to realize is that both models are still limited by the fact that they can only model Gaussian shapes. We can therefore not expect our temporal model to take on shapes that a static model could not. Instead, the advantage of our temporal model is that it can use the additional information given by the neighbourhood-conditional probabilities $p(z_{nk}|\mathrm{MB}(\mathbf{z}_n))$ to more accurately discriminate between classes.

Table 5.2: Classification accuracy of static and temporal models on FCPS datasets.

		Misclassifications	
Dataset	**Size**	**Static**	**Temporal**
Atom	800	294	**0**
Wingnut	1016	36	**8**
Engytime	4096	138	**3**

Chapter 6

Algorithm

In this chapter we present our rare category detection algorithm called Temporal-HMRF based on the temporal model we derive in Chapter 5. First, we apply our algorithm to a simple binary 2D dataset as an illustrative toy example. Afterwards, we discuss the algorithm in more detail. We provide pseudocode for our algorithm as a whole and highlight several important aspects individually. Finally, we summarize some additional properties of our algorithm to allow for easy comparison with other rare category detection algorithms listed in Table 3.1.

6.1 Toy example

We begin our discussion by giving a high-level description of what our algorithm does. It first fits a mixture model to the data. Each mixture component then nominates a number of instances that they consider likely to belong to an undiscovered minority class. An oracle labels all these instances and verifies if we have indeed discovered a new class. If this turns out to be the case we add a new component to our mixture model and assign it to this new class. The remaining nominees that were labelled as one of the classes already known to us are used to improve the parameter estimates for the corresponding components. We repeat this process until we have found at least one instance from every minority class.

We now apply our algorithm to a simple toy example. Figure 6.1 visualizes the process for a synthetic binary 2D dataset. The majority class is made up of two large elongated clusters, while the small cluster in the middle constitutes the minority class that we aim to discover. The ground truth is shown in Figure 6.1a. In order to simulate a temporal pattern the minority instances are grouped sequentially in time, like in our treatment of the FCPS datasets in Section 5.3.

We initially start out with an unlabelled dataset and fit a mixture model to it using the EM algorithm. The blue ellipses in Figure 6.1b show the shapes of the two Gaussian components in our model. Each component then repeatedly nominates one of its instances for labelling by the oracle. The nominees are visualized by a green cross in Figure 6.1c. After eight nominations we find the first instance from the small cluster in the middle and we have discovered the minority class. As soon as we discover a new class we assign it a new component and add it to our mixture model. This new component is visualized in Figure 6.1d by the red contour.

Before we turn to a more detailed discussion of our algorithm we point out some important differences with Interleave, another model-based rare category detection algorithm we discuss in

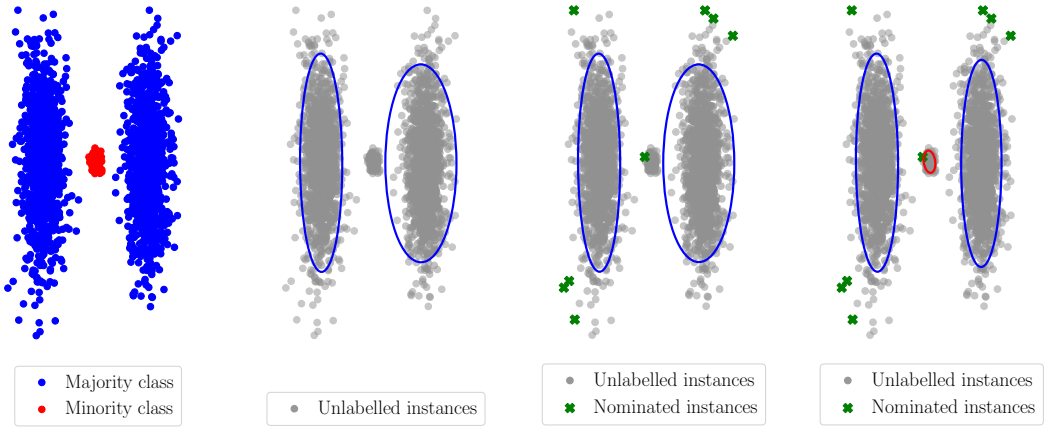


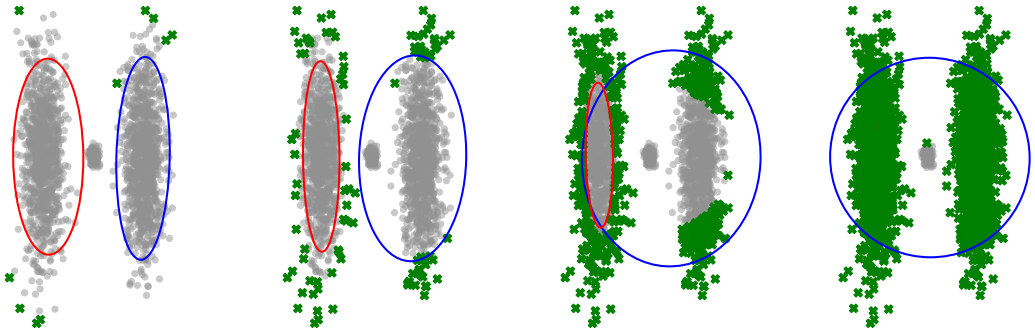
Figure 6.1: Visualization of our algorithm applied on a synthetic binary 2D dataset.

Chapter 3. First, our algorithm uses a temporal model, while Interleave uses a static model. Second, we allow majority classes to be modelled by multiple components, whereas Interleave limits each class to a single component. Third, we only start modelling a minority class once we discover an instance that belongs to it. By contrast, Interleave requires an initial estimate of the number of minority classes and starts out with one component for each class.

We can see why these differences are important by looking at the results of Interleave on the same dataset. Figure 6.2 shows the mixture model that is fitted by the Interleave algorithm after 10, 100, 500, and 1902 labelling requests (out of 2000 instances in total). While the initialization looks similar to our model, observe that there is only a single blue component for the majority class and that the red component is already assigned to the minority class before we know anything about it. As a result, as long as we fail to discover a minority instance, all instances nominated by either component are labelled as belonging to the blue component. Consequently, the red component keeps shrinking while the blue component grows to also incorporate the instances from the left cluster that are being assigned to it. Since the mean of the blue component slowly moves towards the minority class, the model only degrades further over time. It fails to nominate any minority instances until all other instances have already been labelled.

6.2 Detailed description

While the previous toy example gives a good intuitive explanation of our algorithm, we skipped a number of important details and decisions for clarity. We now give a more elaborate treatment of each aspect of our algorithm. Specifically, we distinguish between the following subroutines: (1) initialization, (2) instance selection, (3) feedback processing, and (4) parameter updates. Algorithm 6.1 shows the pseudocode for our Temporal-HMRF algorithm.



(a) (b) (c) (d)

Figure 6.2: Intermediate results of the Interleave algorithm after (a) 10, (b) 100, (c) 500, and (d) 1902 nominations. In each plot we show all instances that have been labelled up until that point as green crosses. The red and blue components are assigned to the minority and majority class, respectively. Only when all 1900 majority instances have already been labelled does Interleave finally find one of the 100 minority instances.

6.2.1 Initialization

When we initially run the algorithm our entire dataset is still unlabelled. Since our goal is to discover minority classes it is preferable that our method does not explicitly require prior information about them. After all, in a realistic setting we may not know the minority classes that are present in the data beforehand. Thus, it makes sense intuitively for our mixture model to only consider classes for which we have proof of existence. Initially, we take these to be just the majority classes that we expect to find.

More formally, we assume each instance $\mathbf{x}_n$ belongs to one of the known classes in $\mathcal{Y}$, which expands over time as new classes are discovered. When we initialize the algorithm we have $\mathcal{Y} = \{-M^-, \ldots, -1\}$, meaning we only consider the majority classes. To simplify our discussion we assume there is only one majority class such that $\mathcal{Y} = \{-1\}$. We then need to specify the number of components for this single majority class, which we denote by by K_0. This means our mixture model initially contains K_0 components, all of which are assigned to class -1 such that $\mathbf{C}_{-1} = \{1, \ldots, K_0\}$. We do not lose any generality by doing so, since we could always split this class up into smaller subclasses that are each modelled by subsets of components.

While our algorithm does not require the number of minority classes as prior information, we do need to manually choose K_0. This choice turns out to be quite important, which we can easily see by considering our previous toy example. Originally, we chose to use $K_0 = 2$, which matches the true number of majority components. However, if we had chosen $K_0 = 1$ instead, the results would have been different. The mean of the single component would have coincided with the minority cluster so that the instances are never nominated as potential outliers. Estimating the number of clusters from the data is still an open problem, though heuristic approaches have been proposed [33, 34].

Algorithm 6.1 Temporal-HMRF

Input: Unlabelled dataset $\mathbf{X}_U$.
Parameters: h, K_0, α.

$\mathcal{Y} \leftarrow \{-1\}$
$K \leftarrow K_0$
$\mathbf{X}_L \leftarrow \emptyset$
$\mathbf{Y}_L \leftarrow \emptyset$
Initialize the mixture model with K_0 components using k-means clustering.
while not all minority classes are discovered **do**
    repeat
        Estimate $\hat{\mathbf{Z}}$ using (5.31), ignoring configurations prohibited by Y_L.
        Compute the responsibilities $\xi(z_{nk})$ or $\xi(z_{nk}|y_n)$ using (5.35) or (5.40), respectively.
        Update the model parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ using (6.5) and (6.6), respectively.
    until convergence
    for $k = 1, \ldots, K$ **do**
        Compile a list $\mathbf{A}_k = \{\mathbf{x}_n \in \mathbf{X}_U | k = \arg\max_{k'} \xi(z_{nk'})\}$.
    $\mathbf{B} \leftarrow \emptyset$
    for $k = 1, \ldots, K$ **do**
        $\mathbf{B} \leftarrow \mathbf{B} \cup \{\arg\max_{\mathbf{x}_n \in \mathbf{A}_k} p(\mathbf{x}_n | z_{nk} = 1)\}$.
    Retrieve the class labels y_n of all $\mathbf{x}_n \in \mathbf{B}$ from the oracle.
    for each $\omega \in \{y_n | \mathbf{x}_n \in \mathbf{B}\}$ such that $\omega \notin \mathcal{Y}$ **do**
        Initialize a new component $K + 1$ with parameters given by (6.1) and (6.2).
        Assign this component to the minority class ω such that $K + 1 \mapsto \omega$.
        $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{\omega\}$
        $K \leftarrow K + 1$
    $\mathbf{X}_U \leftarrow \mathbf{X}_U \setminus \mathbf{B}$
    $\mathbf{X}_L \leftarrow \mathbf{X}_L \cup \mathbf{B}$
    $\mathbf{Y}_L \leftarrow \mathbf{Y}_L \cup \{y_n | \mathbf{x}_n \in \mathbf{B}\}$

6.2.2 Instance selection

We adopt the same strategy as the Interleave algorithm [28] for instance selection. For clarity and completeness we repeat the important details here. We first compile a list $\mathbf{A}_k$ for each component k of unlabelled instances $\mathbf{x}_n \in \mathbf{X}_U$ over which k exerts the most "ownership", i.e. $k = \arg\max_{k'} \xi(z_{nk'})$. Each list is then sorted in ascending order by $p(\mathbf{x}_n | z_{nk} = 1)$. Afterwards, a merged list $\mathbf{B}$ is formed by repeatedly appending the top instance from each individual component to $\mathbf{B}$. Finally, all instances in $\mathbf{B}$ are presented to the oracle for labelling.

Recall that the motivation for this approach in the Interleave algorithm is to eliminate the mixture weights from the equation. While our approach is similar, the motivation is slightly different. After all, our temporal model already eliminates the mixture weights by defining a different distribution over the latent variables given by (5.4). However, this formulation introduces a new problem because it is infeasible to compute the normalization constant Z in practice. This makes it impossible to compute the overall likelihood for each instance. By letting each component nominate instances separately we circumvent this problem.

6.2.3 Feedback processing

Once we discover a new minority class we add a new component $K+1$ to the mixture model and assign it to this class. The choice to model each minority class using just a single component is motivated by the assumption that instances from the same minority class are self-similar and as such form compact clusters in feature space. While we make no assumptions on the exact shape of these clusters we do assume that on a global scale they can be approximated by a single Gaussian.

In order to initialize a new Gaussian component we need to estimate its mean and covariance. Imagine we have just obtained the class label y_n of an instance $\mathbf{x}_n$ that belongs to a new minority class, which we now want to model. Since it is impossible to estimate a covariance matrix from a single instance, we also consider its h "temporal neighbours" that are closest in feature space. We denote the temporal neighbourhood of $\mathbf{x}_n$ by $\mathrm{TN}(\mathbf{x}_n) = \{\mathbf{x}_i | \mathbf{z}_i \in \mathrm{MB}(\mathbf{z}_n)\}$. We then sort the instances $\mathbf{x}_i \in \mathrm{TN}(\mathbf{x}_n)$ by their Euclidean distance to $\mathbf{x}_n$ given by $\|\mathbf{x}_n - \mathbf{x}_i\|$ and select the first h among them. We denote this subset by $\mathrm{TN}_h(\mathbf{x}_n) \subset \mathrm{TN}(\mathbf{x}_n)$.

The intuition behind this approach is that at least h of the temporal neighbours of $\mathbf{x}_n$ belong to the same class (if h is the true neighbourhood radius). Among all $2h$ temporal neighbours it then seems reasonable to select the ones that are closest in feature space. If $h+1 \geq D$ we are generally able to use their sample mean and covariance for initialization such that

$$\boldsymbol{\mu}_{K+1}^{(0)} = \frac{1}{h+1}\left(\mathbf{x}_n + \sum_{\mathbf{x}_i \in \mathrm{TN}_h(\mathbf{x}_n)} \mathbf{x}_i\right), \tag{6.1}$$

$$\boldsymbol{\Sigma}_{K+1}^{(0)} = \frac{1}{h+1}\left((\mathbf{x}_n - \boldsymbol{\mu}_{K+1}^{(0)})(\mathbf{x}_n - \boldsymbol{\mu}_{K+1}^{(0)})^{\mathrm{T}} + \sum_{\mathbf{x}_i \in \mathrm{TN}_h(\mathbf{x}_n)} (\mathbf{x}_i - \boldsymbol{\mu}_{K+1}^{(0)})(\mathbf{x}_i - \boldsymbol{\mu}_{K+1}^{(0)})^{\mathrm{T}}\right). \tag{6.2}$$

If the resulting covariance matrix $\boldsymbol{\Sigma}_{K+1}^{(0)}$ is singular or $h+1 < D$ we use a scaled copy of the global covariance matrix instead. While this might be a poor estimate, these initial inaccuracies can be corrected by subsequent iterations of the EM algorithm. We then obtain

$$\boldsymbol{\mu}_{K+1}^{(0)} = \mathbf{x}_n, \tag{6.3}$$

$$\boldsymbol{\Sigma}_{K+1}^{(0)} = \sigma^2 \boldsymbol{\Sigma}, \tag{6.4}$$

where σ is a small scaling factor (like $\sigma = 0.25$) and $\boldsymbol{\Sigma}$ is the covariance matrix of the whole dataset.

6.2.4 Parameter updates

To update the parameter estimates for each component we have the update rules (5.41) and (5.42). Each iteration, the parameters are updated using all labelled information obtained from previous iterations. There is one issue with these update rules, however. Recall that we initially treat all instances as belonging to a single majority class. As long as we only model a single class there is no added value in labelled feedback. After all, before we discover the first minority class, the oracle can only respond with one possible class label. This is unsatisfactory as we would expect there to be additional value in knowing for sure if an instance belongs to the majority class.

To address this limitation we weigh the contribution of labelled instances so that they have a relatively larger impact on to the parameter updates. Specifically, we modify our update rules (5.41) and (5.42) to incorporate weights λ_ω for instances labelled as class ω. We define separate

weights for each class, because the class size may be a factor in deciding the weight: for larger classes we may want larger weights to ensure each labelled instance has the same relative influence on the parameters in comparison to to a small class. Our modified updated rules are then given by

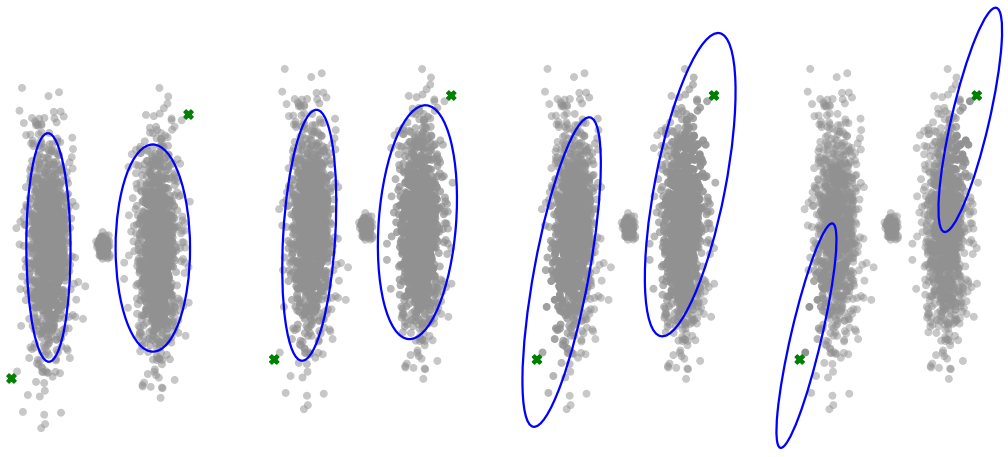
$$\boldsymbol{\mu}_k^{(t+1)} = \frac{\sum_{\mathbf{x}_n \in \mathbf{X}_U} \xi(z_{nk}) \mathbf{x}_n + \sum_{\mathbf{x}_n \in \mathbf{X}_L} \lambda_{y_n} \xi(z_{nk}|y_n) \mathbf{x}_n}{\sum_{\mathbf{x}_n \in \mathbf{X}_U} \xi(z_{nk}) + \sum_{\mathbf{x}_n \in \mathbf{X}_L} \lambda_{y_n} \xi(z_{nk}|y_n)}, \tag{6.5}$$

$$\boldsymbol{\Sigma}_k^{(t+1)} = \frac{\sum_{\mathbf{x}_n \in \mathbf{X}_U} \xi(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}} + \sum_{\mathbf{x}_n \in \mathbf{X}_L} \lambda_{y_n} \xi(z_{nk}|y_n)(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}}{\sum_{\mathbf{x}_n \in \mathbf{X}_U} \xi(z_{nk}) + \sum_{\mathbf{x}_n \in \mathbf{X}_L} \lambda_{y_n} \xi(z_{nk}|y_n)}, \tag{6.6}$$

where the value of each λ_ω is determined by a hyperparameter $0 \leq \alpha < 1$ that specifies the relative contribution of labelled instances to the parameters of components for class ω. We have

$$\lambda_\omega = \max\left(1, \quad \frac{\alpha}{1-\alpha} \cdot \frac{\sum_{\mathbf{z}_n \in \mathbf{Z}_U} \sum_{k \mapsto \omega} \xi(z_{nk})}{\sum_{\mathbf{z}_n \in \mathbf{Z}_L} \sum_{k \mapsto \omega} \xi(z_{nk}|y_n)}\right), \tag{6.7}$$

where the maximum ensures that labelled instances are never valued less than unlabelled instances. The numerator and denominator of the second term represent the "unlabelled" and "labelled" size of ω, respectively. Together, they measure the amount of instances that contribute to the parameters of components for ω. For $\alpha = 0.1$, 10% of the parameters are determined by instances labelled ω.



(a) $\alpha = 0$ (b) $\alpha = 0.1$ (c) $\alpha = 0.5$ (d) $\alpha = 0.9$

Figure 6.3: The effects of various values of the labelled weight ratio α for our toy example after labelling the first two instances, shown here as green crosses.

Choosing an appropriate value for α is a difficult problem. Unlike the neighbourhood radius h there is no direct physical meaning behind α. We do however offer an intuitive interpretation in terms of the relative contribution to the model parameters. To make this more explicit, Figure 6.3 visualizes the effects of different values of α. By setting $\alpha = 0$ we weigh labelled and unlabelled instances equally as in our original model, since we ensure that our effective weights are never smaller than 1.

In the end, the choice comes down to a compromise between model quality and feature space exploration. Larger values cause our model to become more sensitive to previously labelled instances, so we are less likely to nominate similar instances in the future. However, this may also decrease the quality of our model as is clearly the case in Figure 6.3d. Meanwhile, smaller values preserve the accuracy of our model, but since there are not a lot of changes in subsequent iterations we expect to often nominate similar instances. In practice, we recommend to start with a small value of α and increase it during later iterations if it turns out all nominations are highly self-similar.

6.3 Properties

We end our algorithmic discussion by highlighting a few important properties. Some of these have already been mentioned in passing, but it is worthwhile to state them explicitly. We focus on the same properties that we listed in our overview of existing rare category detection algorithms in Table 3.1, namely the necessary assumptions on the data, required prior knowledge, and time complexity.

In addition to the smoothness and compactness assumption, which are shared by all rare category detection algorithms, we make two additional assumptions. First, since we use mixture models, we implicitly assume the minority classes are isolated in feature space. Second, we are focused on temporal data where the class labels are smooth over time, as illustrated earlier in Figure 5.8b. We call this the temporal smoothness assumption.

Strictly speaking our algorithm is prior-free, i.e. it requires no prior knowledge about the data. However, there are a number of hyperparameters that have to be specified manually. These include the neighbourhood radius h, the labelled weight ratio α, and the initial number of component K_0. In Sections 5.3 and 6.2 we already give some insight into the meaning of these hyperparameters. Prior knowledge can also help in choosing appropriate values, especially for h. In Section 7.3 we show how sensitive the performance of our algorithm is to different values.

The time complexity of our algorithm is $\mathcal{O}(DN^2)$, which is the same as similar model-based algorithms like Interleave. The runtime is dominated by the updates of the covariance matrix, which takes $\mathcal{O}(DN^2)$ time. Many programming languages provide access to frameworks that are specifically optimized for vectorized computations such as matrix multiplications. It could therefore be that in practice the ICM algorithm ends up being slower due to its iterative implementation, despite having a linear time complexity.

Chapter 7

Experiments

In this chapter we evaluate and analyse the performance of our algorithm experimentally. Our experiments are motivated by one of our original use cases for temporal rare category detection: finding bird songs in outdoor audio recordings. However, to the best of our knowledge there do not exist any openly available datasets with the proper characteristics, such as class imbalance and segment-level annotations. We therefore generate realistic data ourselves by embedding isolated bird song fragments into existing outdoor audio recordings.

We compare our algorithm against two other rare category detection methods: Interleave, a model-based approach like ours, and NNDM, which uses changes in local density to identify likely minority instances. Both are discussed in more detail in Chapter 3. Additionally, we analyse the contribution of our temporal model compared to a static model in isolation. While Interleave is a static model-based algorithm there are numerous other differences with our algorithm. Any discrepancies in performance are therefore not necessarily explained by our temporal model. For that reason we include an additional algorithm called Static-GMM, which is identical to Temporal-HMRF, except it uses a static model rather than our proposed temporal model.

7.1 Data generation

Before discussing our experiments, we first describe our data generation process in more detail. We know of no existing public bird song dataset for rare category detection, so we instead create our own by embedding bird songs in existing audio recordings. Each combined audio recording then constitutes a single dataset. In order to create realistic data we use existing outdoor recordings as background. These are taken from Freesound[1], a website that hosts a large collection of amateur audio recordings. Table 7.1 shows an overview of the three background recordings that we use.

Our bird songs come from the ICML 2013 bird challenge training dataset [10], which contains recordings of 35 different birds. Originally, each bird song recording has a duration of 30 seconds, but we limit ourselves to excerpts of 3.5 seconds. This is typically enough to capture a full individual bird song. The remainder of each recording often only contains repetitions of the same bird song or silence. Removing most of the silent parts is especially important, because those do not constitute interesting events worth detecting. Additionally, by using short excerpts we obtain larger degrees

[1] `https://freesound.org`

Table 7.1: An overview of the background recordings.

ID	Name	Duration	Number of samples	Sample rate
B1	Storm in Oregon Forest	651 seconds	28,733,300	44100 Hz
B2	Traffic at Five Points	1419 seconds	62,602,740	
B3	Rain, Thunder, Wind Chime, Cars	713 seconds	31,483,400	

Table 7.2: An overview of the bird songs.

ID	Name	Duration	Number of samples	Sample rate
F1	Carduelis Chloris	3.5 seconds	154,350	44100 Hz
F2	Parus Caeruleus			
F3	Phylloscopus Collybita			
F4	Emberiza Citrinella			
F5	Fringilla Coelebs			

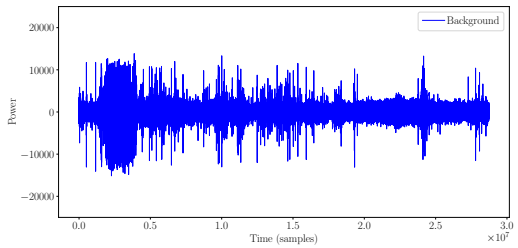
of class imbalance, as the bird songs comprise an even smaller part of the combined recordings. In the end we selected five bird song recordings to use in our experiments, shown in Table 7.2.

We can combine the background recordings and bird songs into either binary or multi-class recordings. Binary recordings are combinations of a background recording with a single bird song. We refer to them by the joint IDs of their ingredients, so "B1F1" refers to the combination of the background B1 and foreground F1. In multi-class recordings we embed all five bird songs in the same background recording at different positions. It is therefore sufficient to specify which background recording is used. We use "B1M" to refer to the multi-class recording of B1 and all five bird songs.

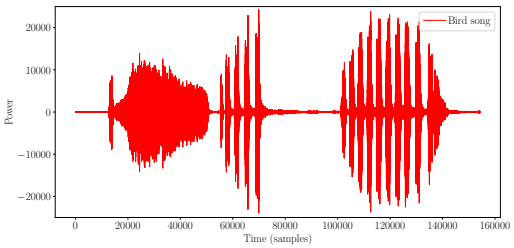
This notation does not specify the position of the bird song in the combined recording. Depending on the position it may be easier or harder to discover the bird song. For example, if we insert the bird song during a period of heavy rain, its sound could be easily obscured. In order to control for this variable we generate 20 different recordings for each combination. In each recording the bird songs are inserted at a randomly chosen position. By averaging the results over all 20 recordings we can reduce the effects of specific positions.

As an example, the raw time signals for the background recording B1 and the bird song F1 are shown in Figure 7.1a and 7.1b, respectively. The bird song contains much fewer samples than the background recording due to its shorter duration. In the case of B1F1 this means the class ratio is approximately 0.0054, meaning the bird song comprises only 0.54% of the combined recording. The severe class imbalance becomes explicit when we consider the raw time signal of B1F1 shown in Figures 7.1c and 7.1d.

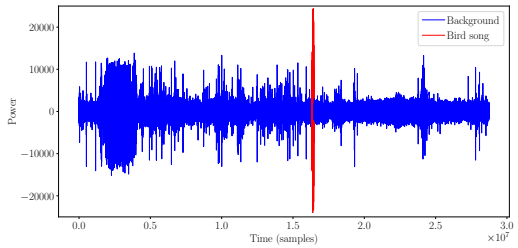
When combining a bird song with the background recording some normalization is necessary. Both recordings come from different sources, so we cannot combine them directly. This can already be seen from Figures 7.1a and 7.1b, where the bird song has a much greater overall power than the



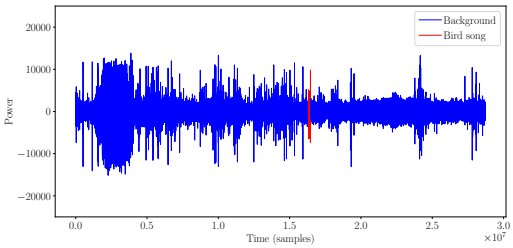
(a) The background recording B1



(b) The bird song F1



(c) A combined recording B1F1 without normalization ($\beta = 1$)



(d) A combined recording B1F1 with normalization ($\beta = 1$)

Figure 7.1: A visualization of the background recording B1, the bird song F1 and a combined binary recording B1F1 (with and without normalization).

background recording. Combining these two signals directly yields the combined signal shown in Figure 7.1c, where the bird song clearly stands out. We therefore first normalize the power of the bird song so that it is equal to the power of the underlying background signal.

Specifically, let us denote the time signal of the background recording by $\mathbf{S}_B = \{s_B^{(t)} | 1 \leq t \leq T_B\}$ and the bird song by $\mathbf{S}_F = \{s_F^{(t)} | 1 \leq t \leq T_F\}$, where T_B and T_F denote the total number of samples in the background recording and bird song, respectively. The normalized time signal of a bird song that is to be inserted at position p is then given by

$$\tilde{s}_F^{(t)} = s_F^{(t)} \cdot \sqrt{\frac{\sum_{t'=1}^{T_F} \left(s_B^{(t'+p-1)} \right)^2}{\sum_{t'=1}^{T_F} \left(s_F^{(t')} \right)^2}}. \tag{7.1}$$

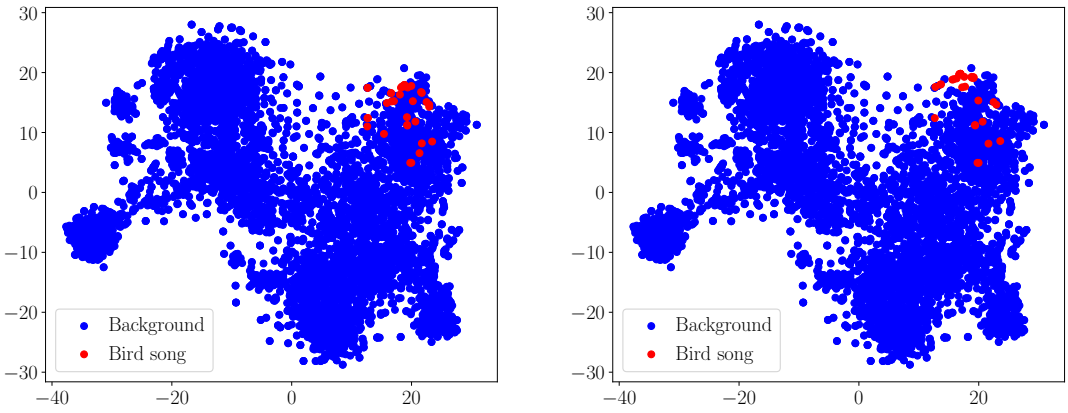
Figure 7.1d shows the result of combining the background recording with the normalized bird song.

This normalization ensures that the bird song has the same average power as the background recording with which it overlaps. However, in reality the bird song may actually be much fainter. For example, the bird could be far away from the recording device or it could be partially obscured by other sounds. To account for this we multiply the normalized bird song signal by a fraction $0 < \beta \leq 1$, which we call the amplification rate. Values of β close to 0 make the bird song barely recognizable, while $\beta = 1$ yields a bird song equal in power to the background. The signal of the

combined recording is then defined as $\mathbf{S} = \left\{ s^{(t)} \;\middle|\; 1 \leq t \leq T \right\}$, where $T = T_B$ and

$$s^{(t)} = \begin{cases} s_B^{(t)} + \beta \cdot \tilde{s}_F^{(t-p+1)} & \text{if } p \leq t < p + T_F, \\ s_B^{(t)} & \text{otherwise.} \end{cases} . \tag{7.2}$$

In order to convert the raw time signal $\mathbf{S}$ to a dataset $\mathbf{X}$ we first split it up into (partially overlapping) windows. Each window corresponds to a single instance in our dataset, and our task is to find the instances that contain bird songs. We use a window length of $w = 0.25$ seconds and a stride of $q = 0.10$ seconds (i.e. subsequent windows are 0.10 seconds apart). For each window we compute $D = 13$ mel-frequency cepstral coefficients (MFCCs) and use them as the feature vector of each instance. MFCCs are commonly used for speech recognition tasks and have previously also been applied to bird song recognition [18]. By mapping the physical frequencies to perceptual frequencies they are meant to more accurately model the human auditory system.



(a) $\beta = 0.1$ (b) $\beta = 1$

Figure 7.2: t-SNE visualizations of B1F1 for $\beta = 0.1$ and $\beta = 1$.

Figure 7.2 shows visualizations of B1F1 for two extreme values of β using t-SNE [21], a popular method for visualizing high-dimensional data. This is a useful way to get an impression of the separability of our data. For $\beta = 0.1$ the bird song seems rather difficult to distinguish from the larger background class. If we increase β we would expect the separability to improve, as the bird song becomes more pronounced. Figure 7.2b suggests that this is indeed the case. Some bird song instances remain indistinguishable, but these most likely correspond to the silent parts of the bird song (the flat regions in Figure 7.1b).

In summary, if we denote the duration of a recording $\mathbf{S}$ by L (in seconds), the corresponding dataset $\mathbf{X}$ is an $N \times D$ matrix with $N = \frac{L-w}{q} + 1$ and $D = 13$. Since $w \ll L$ and $q = 0.10$ we roughly have $N \approx 10L$. Each instance $\mathbf{x}_n \in \mathbf{X}$ is labelled as a minority class if any of its corresponding samples is annotated as a bird song. We treat each unique bird song as a separate minority class, so binary recordings have a single minority class while multi-class recordings have five. All remaining instances that are not labelled as a minority class are part of the background recording, which is represented by a single majority class.

7.2 Binary experiments

We first discuss our experiments on the binary recordings. Because each binary recording contains only one unique bird song our task is simply to find a single minority instance using as few labelling requests as possible. This simplifies the discussion, so we can already develop some intuition for the more involved treatment of multi-class recordings. We use the required number of requests to the labelling oracle as a performance metric for evaluation. Our goal is to minimize this number.

For our algorithm we need to choose values for three hyperparameters: the neighbourhood radius h, the initial number of components K_0, and the labelled weight ratio α. Each bird song has a duration of 3.5 seconds, which roughly corresponds to 36 subsequent instances in our dataset. We therefore choose $h = 18$. We take $\alpha = 0.1$ so that the parameters of each components are determined for 10% by labelled instances. Finally, we use $K_0 = 2$ as an estimate of the initial number of components. This allows our model to remain relatively simple.

We begin our discussion by considering the effects of the amplification rate β on the performance of each algorithm. This gives us an idea of how sensitive each algorithm is to the power of the bird song and what a suitable value might be for further experiments. Figure 7.3 shows the results on B1F1 for $\beta \in \{0.1, 0.2, \ldots, 1.0\}$. As a baseline we include the expected number of requests it would take to find the bird song by random sampling. For binary recordings this value can be computed exactly as $\frac{N+1}{P+1}$, where P denotes the number of bird song (minority) instances [1]. The results are obtained by averaging over 20 recordings, each with a different randomly chosen position of the bird song. Error bars represent the standard error.

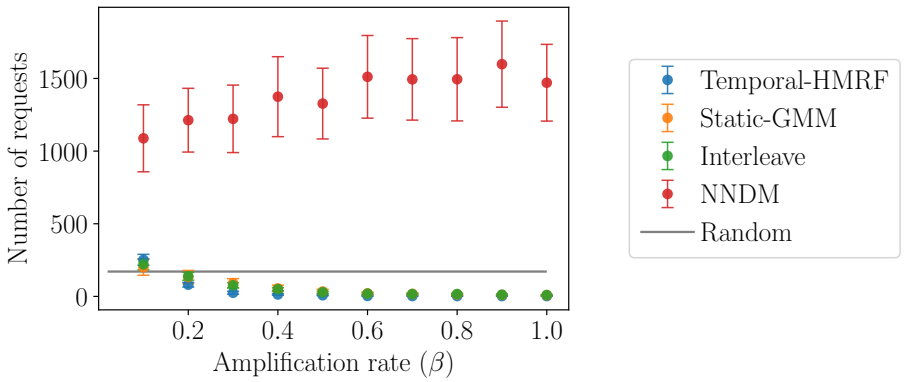


Figure 7.3: The average number of requests required for B1F1 as a function of the amplification rate β.

The first thing we notice from Figure 7.3 is that NNDM consistently performs worse than random sampling. Because NNDM requires so many labelling requests the interesting details for the other algorithms are obscured. We suspect the reason behind its poor performance is because NNDM relies heavily on the compactness and smoothness assumptions. If these assumptions do not hold the strategy of nominating instances from regions with sudden changes in density makes little sense. Indeed, the t-SNE visualizations in Figure 7.2 suggest that these assumptions may not be realistic for our data. Note also that the performance of NNDB does not improve for larger β, since it is designed to work specifically in the non-separable case. As the bird songs become louder

and further separated from the background, nothing meaningful is changed for the strategy used by NNDM. Indeed, the results seem to get slightly worse for larger β.

In the remainder of our discussion we no longer report the results for NNDM. Further preliminary experiments show that its performance is equally poor on other recordings. Including these results in our plots only obscures the more interesting details in other algorithms. Given our plausible explanation for the observed results and the stark difference with the other algorithms we feel this decision is beneficial to our further discussion.

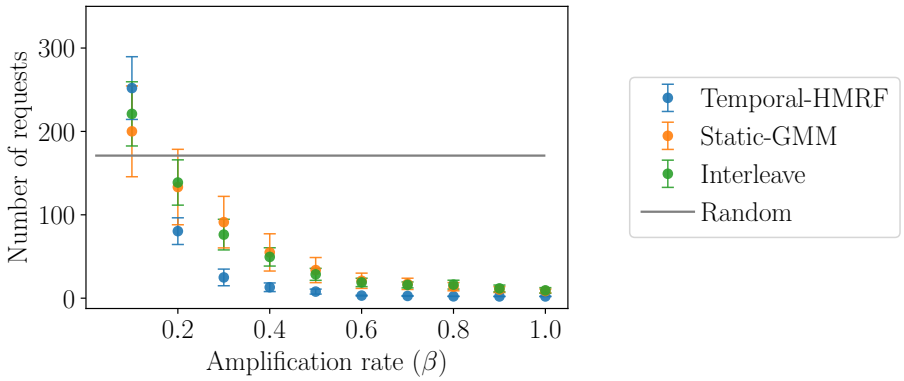


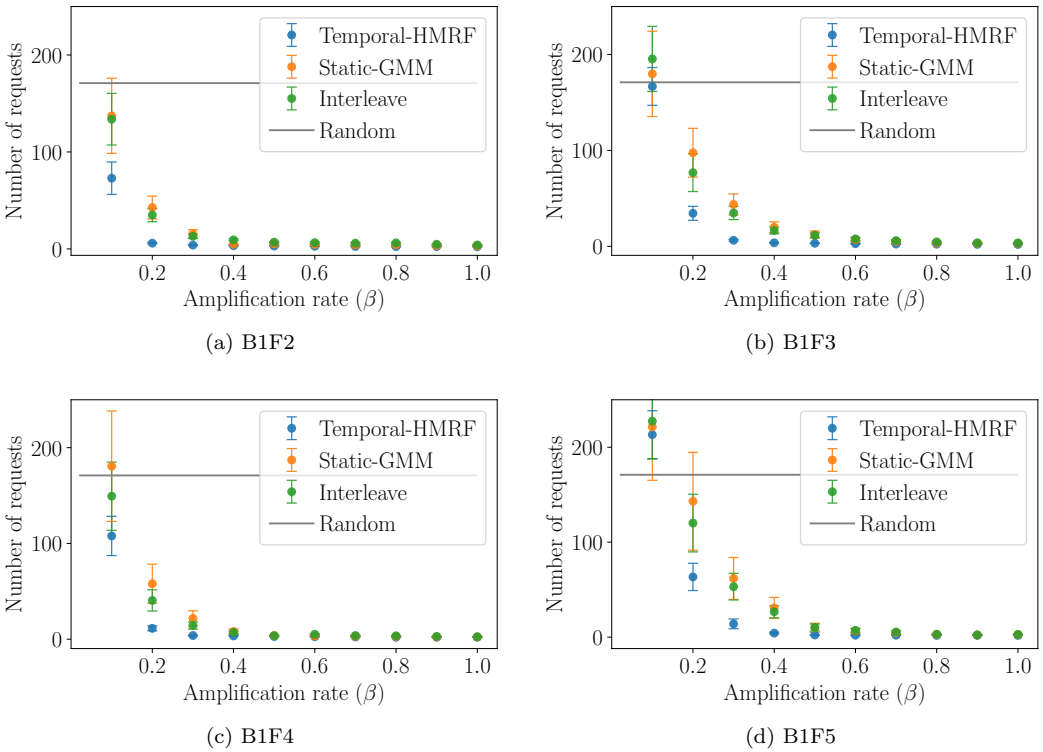
Figure 7.4: The average number of requests required for B1F1 (excluding NNDM) as a function of the amplification rate β.

Figure 7.4 shows the same results for the other algorithms without NNDM. We can make a number of interesting observations based on these results. The performance of each remaining algorithm improves for larger β, and the variance in the results decreases. As the bird songs grow louder they also become more distinguishable from the background. For $\beta = 0.1$ the results of all algorithms are still slightly worse than random. Upon manual inspection this is understandable as B1 contains a lot of rainfall that almost completely drowns out bird songs at this amplification rate. For $\beta > 0.1$ we see all algorithms overtake random sampling as the required number of requests consistently decreases.

Our Temporal-HMRF algorithm outperforms both Static-GMM and Interleave for all $\beta > 0.1$, but for $\beta = 0.1$ it actually has the poorest performance. For very small β it may be that only a single bird song instance meaningfully stands out from the background. Recall that neighbourhood-conditional probabilities play an important role in our temporal model. It could be that such isolated bird song instances are "smoothed away" by neighbouring instances that are wrongfully considered to be part of the majority class. As a result, the parameter updates in our temporal model may be inaccurate, which could explain the degraded performance for $\beta = 0.1$.

Figure 7.5 shows the results for the remaining bird songs F2-F5 in combination with B1. We see that generally the same observations hold. For larger values of β again reduce the required number of requests. Eventually, all algorithms manage to find the bird song with just a single request. For these bird songs our Temporal-HMRF algorithm requires fewer requests even for $\beta = 0.1$, though for B1F3 and B1F5 the difference is negligible.

We now turn to combinations with B2 and B3. Figure 7.6 shows the results for both B2F1 (left) and B3F1 (right). Notice that the range of the y-axes is much smaller than for our earlier



(a) B1F2 (b) B1F3

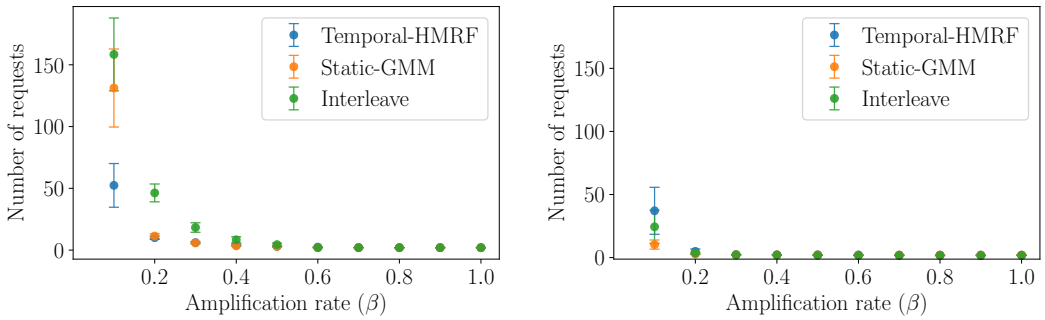
(c) B1F4 (d) B1F5

Figure 7.5: Results for the remaining combinations of B1 with F2-F5 as a function of the amplification rate β.

plot for B1F1. This suggests that it is more difficult to find bird songs in B1 compared to both B2 and B3. Indeed, even for $\beta = 0.1$ all algorithms manage to stay well below the random sampling baseline[2], which is why we no longer show it. For B2F1 our Temporal-HMRF outperforms the other algorithms, but for B3F1 we see that the results are relatively poor for $\beta = 0.1$. This may again be caused by the neighbourhood-conditional probabilities smoothing away isolated bird song instances. To keep the main text uncluttered we show the plots for the remaining bird songs in combination with B2 and B3 in Figure A.1 and A.2 of Appendix A, respectively.

In general, the results for $\beta = 0.1$ and $\beta = 0.2$ are the most interesting to study further. For these values we can still discern some meaningful differences in the performance of each algorithm. For larger values of β we see that the task becomes easy enough for all algorithms to discover the bird song very quickly. Specifically, for B1 we are most interested in $\beta = 0.2$ since the results for $\beta = 0.1$ are typically worse or marginally better than random sampling. This is not the case for B2 and B3, where even for $\beta = 0.1$ the results remain much better than random sampling. In the remainder of this section we therefore limit ourselves to these values of β.

[2]The expected number of requests required by random sampling for combinations involving B2 and B3 equals 373 and 187, respectively. The large difference is due to the longer duration of B2.



(a) B2F1

(b) B3F1

Figure 7.6: Results for B2F1 and B3F1.

Table 7.3 gives a summary of the exact results[3] for each combination and appropriate values of β (i.e. $\beta = 0.1$ for B1, and $\beta = 0.2$ for B2 and B3). Standard deviations are shown in parentheses. Boldfaced entries indicate the best performing algorithm. While these results could already be read from the preceding plots it is easier to compare them across different combinations in this form. The first thing we notice is that the standard deviations are quite large compared to the means. This suggests that the timing of the bird song is an important factor.

We also see that generally F1 is the most difficult bird song to discover regardless of the background recording. However, the background is still the dominant factor overall. Temporal-HMRF consistently outperforms the other two algorithms for B1 and B2, but its performance on B3 is relatively poor. The standard deviations for these recordings are exceptionally large compared to the means, suggesting that these results are mostly the cause of a few outliers.

A limitation of only considering the mean and standard deviation is that we do not obtain any information about potential asymmetries. For a more complete picture we can consider box plots. Figure 7.7 shows box plots for B1F1 ($\beta = 0.2$), B2F1 ($\beta = 0.1$), and B3F1 ($\beta = 0.1$). Here, the boxes extend from the lower to upper quartiles; the line inside represents the median. The whiskers represent the minimum and maximum values within 1.5 times the interquartile range below and above the boxes, respectively. Outliers outside of this range are shown as individual circles. Box plots for the other bird songs F2-F5 in combination with B1, B2, and B3 are shown in Figure A.3, A.4, and A.5 of Appendix A, respectively.

We see that in some cases there is indeed a strong asymmetry in the distribution of the results. Consider for example Static-GMM in Figure 7.7a or Temporal-HMRF in Figure 7.7b. Another important observation is that the variance in the results for Temporal-HMRF is generally smaller than for the other algorithms. In the case of B3F1 we indeed observe that all algorithms are actually similar except for three significant outliers, as we hypothesized earlier.

[3]To assign a more intuitive interpretation to the results, recall that each instance in our dataset represents a window of 0.25 seconds in the original recording. We can thus interpret the number of requests shown in this Table as roughly four times the amount of seconds a human expert has to listen to before finding the bird song. This is slightly unrealistic in a practical setting, as it is unlikely someone could accurately label a 0.25 second extract. However, it works well as an intuitive heuristic.

Table 7.3: The average number of requests required for every possible binary combination. For combinations involving B1 we use $\beta = 0.2$, and for combinations with B2 or B3 we use $\beta = 0.1$. Standard deviations are shown in parentheses.

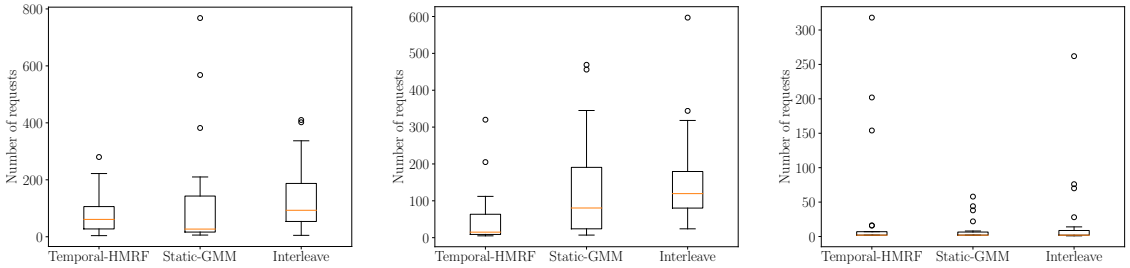
		Temporal-HMRF	Static-GMM	Interleave
$\beta = 0.2$	B1F1	**80.4 (71.6)**	133.3 (202.3)	138.8 (121.5)
	B1F2	**5.9 (3.5)**	42.7 (52.7)	34.9 (30.6)
	B1F3	**34.4 (32.6)**	97.6 (113.9)	76.9 (88.2)
	B1F4	**11.3 (11.4)**	57.9 (91.3)	40.5 (49.7)
	B1F5	**63.5 (64.2)**	143.1 (230.3)	120.2 (135.8)
$\beta = 0.1$	B2F1	**52.4 (79.1)**	131.2 (141.3)	158.4 (131.5)
	B2F2	9.75 (12.5)	**8.1 (12.5)**	15.7 (15.7)
	B2F3	**5.75 (2.2)**	11.0 (11.9)	31.3 (28.2)
	B2F4	**4.6 (2.7)**	5.0 (3.5)	12.8 (16.5)
	B2F5	**11.6 (14.1)**	13.8 (18.1)	33.3 (38.3)
$\beta = 0.1$	B3F1	37.1 (83.3)	**10.3 (16.2)**	24.4 (58.5)
	B3F2	**2.4 (0.8)**	2.5 (1.1)	2.9 (2.0)
	B3F3	5.3 (10.1)	**3.0 (1.7)**	3.4 (3.0)
	B3F4	5.3 (10.3)	**2.8 (1.7)**	3.3 (2.6)
	B3F5	12.0 (35.3)	**4.1 (4.1)**	5.6 (7.9)

7.3 Multi-class experiments

A limitation of the experiments on binary recordings is that we cannot effectively gauge the quality of our temporal model on the minority classes. After all, as soon as we find a single minority instance the algorithm is terminated. We thus never have to actually model a minority class. From that perspective the multi-class recordings are more interesting. Here, we explicitly incorporate newly discovered minority classes into our model until we have discovered all of them.

We can still use the insights we gain from the binary recordings to simplify our discussion of the multi-class recordings. For example, we have seen that typically all model-based algorithms manage to perform well for larger values of β. The most interesting recordings to analyse usually have $\beta = 0.1$ or $\beta = 0.2$. For our multi-class experiments we therefore focus exclusively on these. Table 7.4 shows the average required number of requests to discover all bird songs in each background recording for both $\beta = 0.1$ and $\beta = 0.2$. Standard deviations are again shown in parentheses. The corresponding box plots are shown in Figure 7.8.

Table 7.4 and Figure 7.8 largely show the same phenomena. As we would expect, each algorithm requires more requests for the multi-class recordings compared to any of the binary recordings. Also, the performance of each algorithm still improves when we compare the results for $\beta = 0.1$ with



(a) B1F1 ($\beta = 0.2$) (b) B2F1 ($\beta = 0.1$) (c) B3F1 ($\beta = 0.1$)

Figure 7.7: Box plots for B1F1 ($\beta = 0.2$), B2F1 ($\beta = 0.1$), and B3F1 ($\beta = 0.1$).

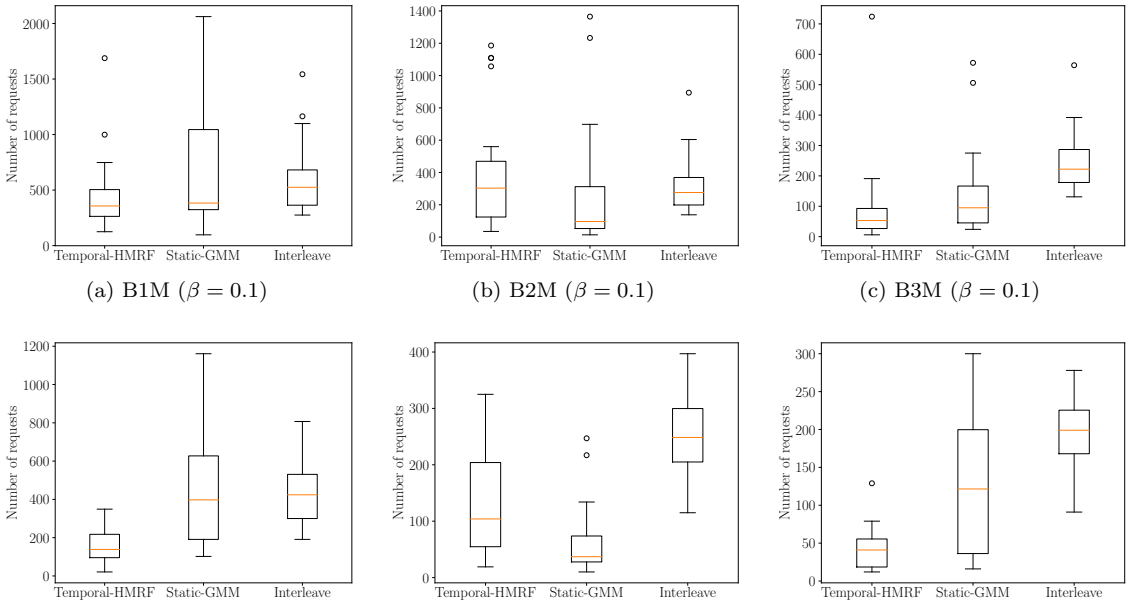
Table 7.4: The average number of requests required for every possible multi-class combination. Standard deviations are shown in parentheses.

		Temporal-HMRF	Static-GMM	Interleave
$\beta = 0.1$	B1M	**452.2 (349.0)**	636.7 (510.8)	614.5 (329.4)
	B2M	417.7 (377.8)	**282.9 (385.8)**	333.2 (182.8)
	B3M	**96.2 (150.2)**	146.4 (148.8)	248.1 (102.4)
$\beta = 0.2$	B1M	**159.4 (90.5)**	450.6 (291.2)	437.0 (160.6)
	B2M	137.8 (105.4)	**64.9 (64.8)**	250.9 (71.9)
	B3M	**43.0 (28.2)**	126.2 (87.9)	193.95 (50.6)

$\beta = 0.2$. For both B1M and B3M our Temporal-HMRF algorithm outperforms the other algorithms, but B2M seems to be troublesome. Though there is a notable difference between $\beta = 0.1$ and $\beta = 0.2$, the Static-GMM algorithm is still better in both cases. This is interesting because our algorithm has no trouble finding the individual bird songs in the binary combinations with B2.

To see what is happening we consider a single trial for one of the B2M recordings as an example. Figure 7.9 shows a learning curve for one such trial, where we plot the number of discovered minority classes against the number of requests. These curves can give more insight into the process of a rare category detection algorithm. We see that both Temporal-HMRF and Static-GMM manage to quickly find four bird songs, but it takes Temporal-HMRF much longer to find the fifth. It appears as if after incorporating the first four minority classes in our model the fifth bird song becomes more difficult to discover.

The observation that Temporal-HMRF generally shows bigger improvements between $\beta = 0.1$ and $\beta = 0.2$ than Static-GMM and Interleave is consistent with our previous hypothesis: in our binary experiments we also noticed that Temporal-HMRF is sometimes outperformed for smaller β. We assume that this is because for smaller β single bird song instances may be smoothed away by their wrongly classified neighbours. This also happens to some extent in the multi-class recordings. Here, for $\beta = 0.1$ the Interleave algorithm outperforms Temporal-HMRF, but for $\beta = 0.2$ the situation is reversed.



(a) B1M ($\beta = 0.1$) (b) B2M ($\beta = 0.1$) (c) B3M ($\beta = 0.1$)

(d) B1M ($\beta = 0.2$) (e) B2M ($\beta = 0.2$) (f) B3M ($\beta = 0.2$)

Figure 7.8: Box plots of the results for B1M, B2M, and B3M for $\beta = 0.1$ and $\beta = 0.2$.

In the remainder of this section we establish how sensitive our algorithm is to its hyperparameters. Recall that in the preceding experiments we use $h = 18$, $\alpha = 0.1$ and $K_0 = 2$. We now repeat the multi-class experiments for $\beta = 0.2$ with different values for each of these hyperparameters and compare them to our original results. We verify whether our algorithm is robust against small changes in the hyperparameters, and determine the effects of more extreme variations. Specifically, we include $h \in \{15, 18, 20, 50\}$, $\alpha \in \{0, 0.1, 0.2, 0.5, 0.7\}$ and $K_0 \in \{1, 2, 3, 4\}$ in our experiments. We only focus on recordings with $\beta = 0.2$ because these require fewer requests overall and are therefore faster to experiment with. Since we are mostly interested in determining the difference in performance compared to our original hyperparameters, this is sufficient.

Table 7.5 shows the results for different values of the neighbourhood radius h. It is clear that $h = 50$ leads to the worst performance overall. This is easily explained by the fact that $h = 50$ does not remotely represent the "true" neighbourhood radius. The results for smaller variations in h are closer together, though $h = 20$ is already worse by some margin for B1M and B2M. This demonstrates that our algorithm is decently robust to small variations in h.[4] This is desirable since in practice there may often be some uncertainty about its true value.

Table 7.6 shows the results for various labelled weight ratios α. A value of $\alpha = 0$ means we always weigh the labelled and unlabelled instances equally (i.e. $\lambda_\omega = 1$ for all $\omega \in \mathcal{Y}$), since we make sure the weights λ_ω are never smaller than 1. We see that it is indeed beneficial to emphasize

[4] We did not evaluate the performance for $h \ll 15$. This is because our component initialization strategy requires $h + 1 \geq D = 13$ (as explained in Section 6.2). For smaller h we would have to use an alternative strategy to initialize new components and the comparison would no longer be fair.

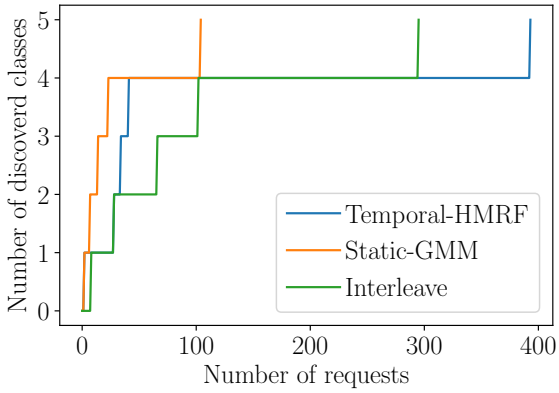


Figure 7.9: One of the learning curves corresponding to a single recording of B2M ($\beta = 0.1$).

Table 7.5: Results for different values of the neighbourhood radius h on all multi-class recordings for $\beta = 0.2$. Standard deviations are shown in parentheses.

	$h = 15$	$h = 18$	$h = 20$	$h = 50$
B1M	164.3 (78.9)	**159.4 (90.5)**	170.7 (93.1)	297.8 (187.0)
B2M	**131.4 (108.2)**	137.8 (105.4)	160.9 (110.9)	258.5 (157.2)
B3M	47.5 (28.0)	43.0 (28.2)	**38.0 (27.3)**	85.3 (45.1)

the labelled instances as the performance generally increases for non-zero values of α. Even for small values such as $\alpha = 0.1$ and $\alpha = 0.2$ the effects are quite noticeable. We observe the most interesting results for $\alpha = 0.5$. For B1M the results degrade a lot, but for B2M and B3M they notably improve. Intuitively, this matches the volatile behaviour associated with larger values of α as depicted in Figure 6.3. For $\alpha = 0.7$ the results start to degrade for B3M as well, though they continue to improve for B2M. While we are unsure of the exact reason this happens, it could be used to address the relatively poor performance of Temporal-HMRF on B2M that we noted earlier.

Broadly speaking we can conclude two things from these results, though further experiments are necessary to verify whether these conclusions hold more generally. First, the optimal value of α largely depends on the data and is therefore difficult to find beforehand. Second, using small values for α seems to be a stable option overall. This reinforces our earlier suggestion of starting out with a small value and gradually increasing it if necessary.

Finally, Table 7.7 lists the results of initializing the algorithm with a varying number of components. In general, our chosen value of $K_0 = 2$ seems to be a consistent and safe option. For B1M and B2M this yields the best results and for B3M it is still easily the second best option. This illustrates the various considerations that go into choosing K_0. For larger values our model complexity grows and it should in theory be able to model more complex data distributions. However, on the other hand this also means that we nominate more instances during each iteration. If some of the components continuously nominate uninteresting instances this may eventually result

Table 7.6: Results for different values of the labelled weight ratio α on all multi-class recordings for $\beta = 0.2$. Standard deviations are shown in parentheses.

	$\alpha = 0$	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.5$	$\alpha = 0.7$
B1M	164.5 (77.6)	159.4 (90.5)	**150.6 (98.3)**	198.25 (123.7)	256.3 (153.7)
B2M	158.0 (110.0)	137.8 (105.4)	122.2 (96.9)	74.3 (52.5)	**61.8 (31.0)**
B3M	42.8 (25.4)	43.0 (28.2)	44.2 (30.1)	**33.7 (21.1)**	43.7 (35.9)

in worse performance. Furthermore, adding more initial components to the model introduces the risk of accurately modelling the minority classes from the start, meaning they are less likely to be nominated. We can circumvent these problems by initially using just a single component, but this likely yields a poor model the entire majority class. On these recordings choosing $K_0 = 2$ seems like a safe middle ground, but there is no guarantee this also holds for other kinds of data.

Table 7.7: Results for a varying number of initial components K_0 on all multi-class recordings for $\beta = 0.2$. Standard deviations are shown in parentheses.

	$K_0 = 1$	$K_0 = 2$	$K_0 = 3$	$K_0 = 4$
B1M	262.8 (179.2)	**159.4 (90.5)**	205.9 (108.3)	226.0 (116.3)
B2M	212.1 (150.9)	137.8 (105.4)	**134.9 (115.6)**	181.6 (124.6)
B3M	**31.9 (21.7)**	43.0 (28.2)	59.9 (35.9)	81.1 (43.0)

7.4 Informal experiments on real data

We close this chapter with a brief informal discussion of the behaviour of our algorithm on some natural bird song recordings. The data was collected in 2014 over the span of a week using two recording devices placed in a jungle near Manizales, Colombia. Each device was encased in a metal structure fixed to a tree in order to protect them from the sun and heavy tropical rains. One of them recorded continuously, while the other recorded for 15 minutes per hour. Both devices have a sampling rate of 16 kHz. Figure 7.10 shows a photograph of one of the installed devices.

Because the data has not been annotated we cannot perform a large-scale quantitative analysis. Our intention is therefore not to report the performance on these recordings as further justification for our algorithm. Instead, we describe the sort of sounds that our algorithm nominates to give some examples of what to expect from natural data. These examples are mostly anecdotal, which is why we emphasize that this is an informal discussion.

We focus on four recordings from two different days listed in Table 7.8. Each recording comes from the second recording device, so they all have a duration of 15 minutes. For each recording we let our algorithm nominate 20 instances which we listen to and characterize ourselves. We use the same hyperparameters as before, i.e. $h = 18$, $K_0 = 2$, and $\alpha = 0.1$.

The first recording is very quiet overall and includes almost no noisy background sounds, so the

Figure 7.10: One of the recording devices encased in a metal structure.

bird songs clearly stand out. Indeed, all of the early nominations by our algorithm correspond to bird songs. It is interesting to note that our algorithm rarely nominates the same bird song twice. While this experiment is far from conclusive it could be an indication that each newly discovered bird song is accurately incorporated into the model.

In the second recording we hear a radio playing in the background and occasionally people are talking. These sounds come from a nearby ecological reserve. Despite the noise our algorithm still almost exclusively nominates bird songs, even if they overlap with music in the background. We suspect that this is because the background sounds are mostly low frequency, while the bird songs are high frequency. The MFCCs we use as features are well-suited for distinguishing between them.

The third recording was made during the night, so there were very few birds singing overall. We could barely make out a very faint bird song that lasted for a few seconds ourselves. However, our algorithm fails to find it, because the recording is dominated by continuous heavy rainfall that causes a loud clatter on the metal structures protecting the recording devices.

The fourth and final recording shows many similarities with the first two. There are again some noisy background sounds such as a car alarm going off, an aircraft flying over and a dog barking. Overall, the bird songs still stand out clearly so our algorithm has no trouble finding them.

Table 7.8: An overview of bird song recordings from a jungle near Manizales, Colombia.

ID	**Date**	**Time**	**Number of samples**	**Duration**
R1	04-19-2014	1PM	14,400,128	900 seconds
R2	04-19-2014	5PM		
R3	04-20-2014	3AM		
R4	04-20-2014	12PM		

Chapter 8

Discussion

In the preceding chapters we show the strengths of our temporal model and algorithm by highlighting the advantages over their static counterparts. Section 5.3 demonstrates that our model can achieve higher classification accuracy on several synthetic datasets. The experimental results presented in Chapter 7 show that Temporal-HMRF manages to outperform static rare category detection algorithms for several bird song detection tasks. However, it is equally important to discuss the limitations of our algorithm and identify potential weaknesses. Afterwards, we also highlight promising future research directions for temporal rare category detection.

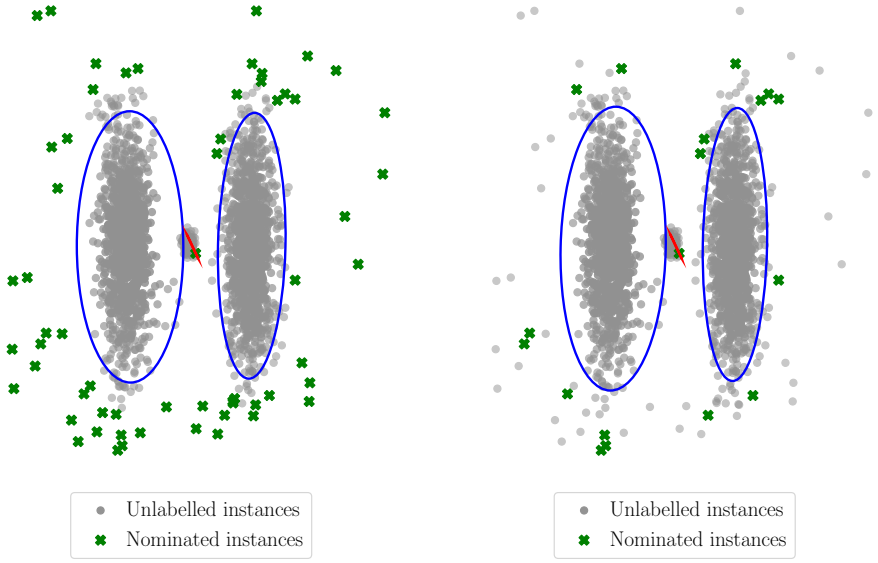
8.1 Limitations

First, there are a number of hyperparameters that our algorithm requires as manual input. Specifically, the user needs to specify the neighbourhood radius h, the labelled weight ratio α, and the initial number of components K_0. While we discuss several useful heuristics in Section 5.3 and 6.2, these are not guaranteed to work. In Section 7.3 we show the effects of different parameter settings on the performance of our algorithm. While our algorithm appears to be robust to small variations in h and K_0, it remains difficult to estimate good values for α.

Our model is also limited by the fact that the neighbourhood radius h is the same for all classes. In reality it may be so that some of the classes have unique temporal structures and therefore require separate definitions of their temporal neighbourhoods. Currently, this cannot be incorporated in our model.

Like similar model-based rare category detection algorithms we require that the minority classes are separable from the majority class. Gaussian mixture models are generally unable to effectively account for overlap in classes. In Section 5.3 we show that for the Engytime dataset our model can successfully classify instances even if they reside deep inside the other class. However, this is only possible if the overlapping region is relatively small. Our algorithm cannot efficiently discover minority classes that overlap completely with the majority class.

Our algorithm is best suited for data with reasonably small dimensionality. There are two important reasons for why high-dimensional data may prove troublesome. First, our algorithm has to estimate full covariance matrices for all components. This means the number of parameters for each component scales quadratically with the dimensionality of the dataset. Second, when initializing a new component we use h temporal neighbours to compute an initial sample covariance



(a) Without exemption

(b) With exemption

Figure 8.1: The effect of noise on our algorithm.

matrix. This requires that $h+1 \geq D$, meaning a higher dimensionality poses additional restrictions on the neighbourhood radius h.

Finally, our algorithm can be sensitive to noise. If many anomalous noisy instances are present in the data we might repeatedly nominate them before identifying the more interesting minority instances. This is a common problem in rare category detection, but some solutions have been proposed. For example, [16] proposes to identify isolated instances as a pre-processing step to exempt them from being nominated. The idea is that minority classes are locally compact, while noise is essentially random and therefore different from other noisy instances and normal data.

Figure 8.1 shows what happens when we add noise to our toy example from Section 6.1. Without an exemption list we nominate many of the noisy instances before discovering the minority class in the centre. By using an exemption list many of these noisy instances can be ignored. However, there are also disadvantages to this approach. First, it makes the algorithm more dependent on the compactness assumption. As evidenced by NNDM, this could have detrimental consequences for the performance. If the minority class is not compact enough we may exempt many genuine minority instances from nomination and never find them. It also requires manual configuration of an additional hyperparameter that decides when an instance is considered isolated.

8.2 Future research

Our proposed model is a first step towards incorporating long-term temporal information into models of our data. In this thesis we demonstrate the effectiveness of our temporal model and algorithm, but also invite several other promising research directions. For example, in Section 5.2 we mention the possibility of using different weight distributions for neighbouring latent variables. While we briefly mention a few potential weight distributions, it would be interesting to analyse the possibilities and their consequences in more detail.

One of the strengths of MRFs is that we are free to define the energy function ourselves. In this thesis we use a relatively simple definition given by (5.9). However, our formulation can easily be extended. This makes it possible to incorporate additional prior information directly into our model. For example, currently we do not explicitly encode the knowledge that our classes are severely imbalanced. In our energy function we could introduce additional terms that emphasize instances belonging to a minority class.

While discussing the motivation for our model we mention that MRFs are traditionally used for spatially smooth data. Since our model is based heavily on MRFs it may be possible to apply it to spatial data as well. Even though our thesis focuses specifically on temporal data our formulation of neighbourhoods could easily be extended to include spatial data. This could allow our algorithm to also be applied to different problem domains such as image segmentation, where the labels of neighbouring pixels also tend to be similar.

Our algorithm requires an initial estimate of the number of clusters in the data, which is still an open problem. It is unrealistic to expect this problem to be solved completely, since the definition of what constitutes a cluster depends on the data and the application. However, novel heuristic approaches such as [33, 34] could still be applicable to many different datasets.

A related research question is how to define the trade-off between the initial number of components and the required amount of nominations. A larger number of components leads to a more complex model for the majority of the data. In theory this could make the minority classes stand out more clearly. At the same time, it may be that some initial components start out modelling a minority class, which makes it more difficult for our algorithm to discover them. Additionally, since each component nominates one instance during each iteration, a larger number of components also results in more nominations per iteration.

One of the limitations of our algorithm has to do with the initialization of new components. Currently, we use the temporal neighbourhood of a newly discovered minority instance for an initial estimate of the covariance matrix. However, this heuristic is quite simplistic and could be improved upon. For example, there is some research on greedy EM algorithms that also introduce new components during execution [37, 38]. Our algorithm could benefit from similar methods if they decouple the neighbourhood radius h from the dimensionality D.

Finally, it may sometimes be necessary to split up the entire data collection into smaller subsets. For example, observatories typically record a large amount of data each day. Due to the sheer volume it may be infeasible to run our algorithm on data spanning an entire year or more. This opens up several new challenges. First, some of the smaller subsets of the data may not contain any interesting minority classes at all. In those cases it would be desirable if our algorithm could at some point classify the entire dataset as "normal" to prevent wasteful nominations. Second, we could investigate how to share valuable insights gained from one of the earlier datasets and use those to our advantage when analysing subsequent datasets.

Chapter 9

Conclusion

Many applications are characterized by severe class imbalance where the minority classes are of primary interest to us. Existing rare category detection algorithms focus almost exclusively on static data in which instances are assumed to be independent. However, there are often clear temporal patterns that cause instances to be strongly correlated. We focus specifically on data where the class labels of instances are smooth over time.

In this thesis we propose a temporal mixture model based on hidden Markov random fields. Our model is able to explicitly incorporate the temporal smoothness of the data. As a result, it no longer recognizes the mixture weights that are prominent in static mixture models. Instead, we formulate neighbourhood-conditional probabilities for the latent variables conditioned on the states of their neighbours. We derive semi-supervised parameter update rules for our novel formulation through the expectation-maximization framework.

Compared to static mixture models our temporal model has two important advantages. First, it is able to improve the parameter estimates by making use of the additional temporal information. Second, it is more robust to class overlap and can correctly classify instances that reside deep inside the boundaries of another class. We demonstrate these advantages on a number of synthetic clustering datasets in Section 5.3.

Based on our temporal model we present a rare category detection called Temporal-HMRF. By fitting our temporal model to the data it learns a more accurate description of the data. Our algorithm then nominates anomalous instances it believes are likely candidates for interesting minority instances. We evaluate its performance on several amateur outdoor audio recordings in which we artificially embed bird songs. We show that our algorithm can outperform existing static rare category detection methods for this type of data.

One of the limitations of our algorithm is that it requires numerous hyperparameters to be specified manually. We verify experimentally that our algorithm generally appears to be robust to small variations, though more extreme values may have a large impact on the performance. Other important limitations to consider include the sensitivity to noisy instances and the fact that our algorithm may not be suitable for high-dimensional data.

Due to its flexible formulation our model can be extended in several ways, providing various promising directions for future research. For example, additional prior information can be incorporated into the energy function of the MRF. We also expect our model to be equally applicable to spatial data after appropriately updating the definition of the neighbourhoods.

Bibliography

[1] John Ahlgren. "The Probability Distribution for Draws Until First Success Without Replacement". In: *arXiv preprint arXiv:1404.1161* (2014).

[2] Julian Besag. "On the statistical analysis of dirty pictures". In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1986), pp. 259–302.

[3] Julian Besag. "Spatial interaction and the statistical analysis of lattice systems". In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1974), pp. 192–236.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN: 0387310738.

[5] Moritz Blume. "Expectation maximization: A gentle introduction". In: *Technical University of Munich Institute for Computer Science* (2002).

[6] Charles Bouveyron, Stéphane Girard, and Cordelia Schmid. "High-dimensional data clustering". In: *Computational Statistics & Data Analysis* 52.1 (2007), pp. 502–519.

[7] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". In: *ACM computing surveys (CSUR)* 41.3 (2009), p. 15.

[8] Shubhomoy Das et al. "Incorporating expert feedback into active anomaly discovery". In: *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE. 2016, pp. 853–858.

[9] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the royal statistical society. Series B (methodological)* (1977), pp. 1–38.

[10] F Deroussen and F Jiguet. "La sonothèque du Muséum: Oiseaux de France, les passereaux". In: *Nashvert production, Charenton, France* (2006).

[11] Haibo He and Edwardo A Garcia. "Learning from imbalanced data". In: *IEEE Transactions on knowledge and data engineering* 21.9 (2009), pp. 1263–1284.

[12] Jingrui He and Jaime Carbonell. "Prior-free rare category detection". In: *Proceedings of the 2009 SIAM International Conference on Data Mining*. SIAM. 2009, pp. 155–163.

[13] Jingrui He and Jaime G Carbonell. "Nearest-neighbor-based active learning for rare category detection". In: *Advances in neural information processing systems*. 2008, pp. 633–640.

[14] Jingrui He, Yan Liu, and Richard Lawrence. "Graph-based rare category detection". In: *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE. 2008, pp. 833–838.

[15] Hao Huang et al. “CLOVER: a faster prior-free approach to rare-category detection”. In: *Knowledge and Information Systems* 35.3 (2013), pp. 713–736.

[16] Hao Huang et al. “RADAR: Rare category detection via computation of boundary degree”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2011, pp. 258–269.

[17] Xuedong D Huang, Yasuo Ariki, and Mervyn A Jack. “Hidden Markov models for speech recognition”. In: (1990).

[18] Chang-Hsing Lee, Yeuan-Kuen Lee, and Ren-Zhuang Huang. “Automatic recognition of bird songs using cepstral coefficients”. In: *Journal of Information Technology and Applications* 1.1 (2006), pp. 17–23.

[19] Na Li and Matthew Stephens. “Modeling linkage disequilibrium and identifying recombination hotspots using single-nucleotide polymorphism data”. In: *Genetics* 165.4 (2003), pp. 2213–2233.

[20] Zhenguang Liu et al. “Prior-free rare category detection: More effective and efficient solutions”. In: *Expert Systems with Applications* 41.17 (2014), pp. 7691–7706.

[21] Laurens van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE”. In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

[22] James MacQueen et al. “Some methods for classification and analysis of multivariate observations”. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297.

[23] David J Miller and Hasan S Uyar. “A mixture of experts classifier with learning based on both labelled and unlabelled data”. In: *Advances in neural information processing systems*. 1997, pp. 571–577.

[24] Todd K Moon. “The expectation-maximization algorithm”. In: *IEEE Signal processing magazine* 13.6 (1996), pp. 47–60.

[25] Kamal Nigam et al. “Text classification from labeled and unlabeled documents using EM”. In: *Machine learning* 39.2-3 (2000), pp. 103–134.

[26] Animesh Patcha and Jung-Min Park. “An overview of anomaly detection techniques: Existing solutions and latest technological trends”. In: *Computer networks* 51.12 (2007), pp. 3448–3470.

[27] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

[28] Dan Pelleg and Andrew W Moore. “Active learning for anomaly and rare-category detection”. In: *Advances in neural information processing systems*. 2005, pp. 1073–1080.

[29] Johan A du Preez. “Efficient training of high-order hidden Markov models using first-order representations”. In: *Computer speech & language* 12.1 (1998), pp. 23–39.

[30] Lawrence R Rabiner and Biing-Hwang Juang. “An introduction to hidden Markov models”. In: *ieee assp magazine* 3.1 (1986), pp. 4–16.

[31] Michael Seifert et al. “Parsimonious higher-order hidden Markov models for improved array-CGH analysis with applications to Arabidopsis thaliana”. In: *PLoS computational biology* 8.1 (2012), e1002286.

[32] Burr Settles. “Active learning”. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6.1 (2012), pp. 1–114.

[33] Catherine A Sugar and Gareth M James. “Finding the number of clusters in a dataset: An information-theoretic approach”. In: *Journal of the American Statistical Association* 98.463 (2003), pp. 750–763.

[34] Robert Tibshirani, Guenther Walther, and Trevor Hastie. “Estimating the number of clusters in a data set via the gap statistic”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63.2 (2001), pp. 411–423.

[35] Alfred Ultsch. “Clustering wih som: U* c”. In: *Proceedings of the 5th Workshop on Self-Organizing Maps*. Vol. 2. 2005, pp. 75–82.

[36] Pavan Vatturi and Weng-Keen Wong. “Category detection using hierarchical mean shift”. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2009, pp. 847–856.

[37] Jakob J Verbeek, Nikos Vlassis, and Ben Kröse. “Efficient greedy learning of Gaussian mixture models”. In: *Neural computation* 15.2 (2003), pp. 469–485.

[38] Nikos Vlassis and Aristidis Likas. “A greedy EM algorithm for Gaussian mixture learning”. In: *Neural processing letters* 15.1 (2002), pp. 77–87.

[39] Yongyue Zhang, Michael Brady, and Stephen Smith. “Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm”. In: *IEEE transactions on medical imaging* 20.1 (2001), pp. 45–57.

[40] Dawei Zhou et al. “Bi-level rare temporal pattern detection”. In: *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE. 2016, pp. 719–728.

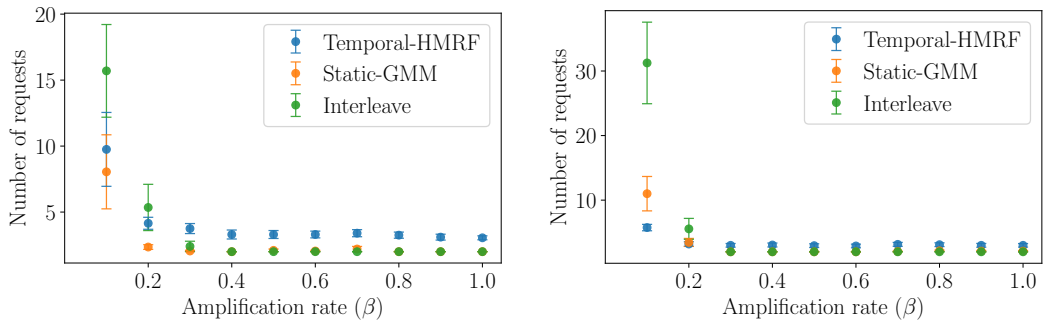
[41] Dawei Zhou et al. “Rare category detection on time-evolving graphs”. In: *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE. 2015, pp. 1135–1140.

Appendices

Appendix A

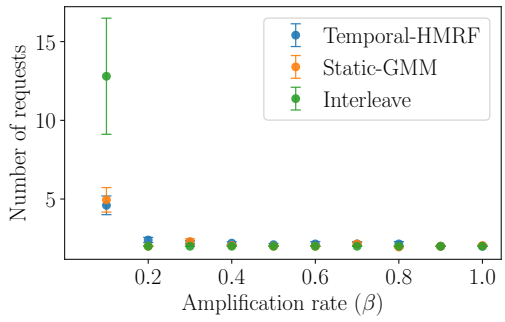
Results for binary recordings

This appendix contains additional Figures for the experiments on binary recordings. Similar Figures are included in the main discussion of Section 7.2, so it is not crucial to show these additional Figures there as well. For the sake of completeness we include them here instead.

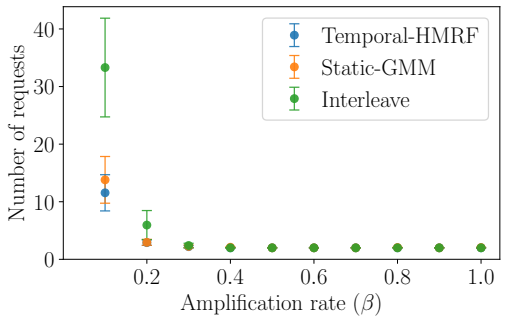


(a) B2F2

(b) B2F3

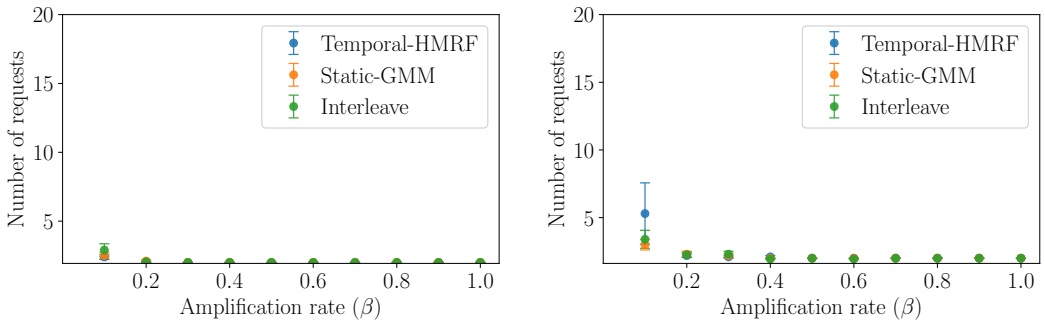


(c) B2F4



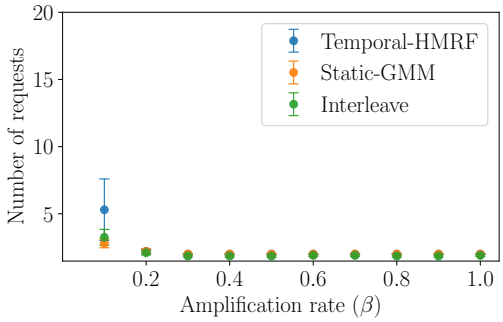
(d) B2F5

Figure A.1: Results for F2-F5 with B2 as a function of the amplification rate β.

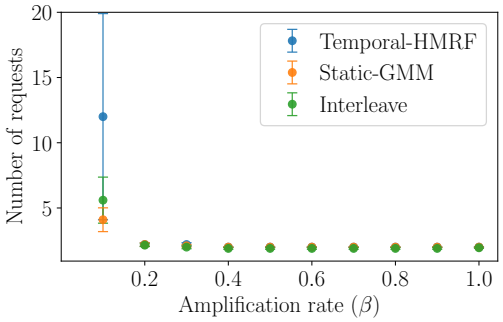


(a) B3F2

(b) B3F3

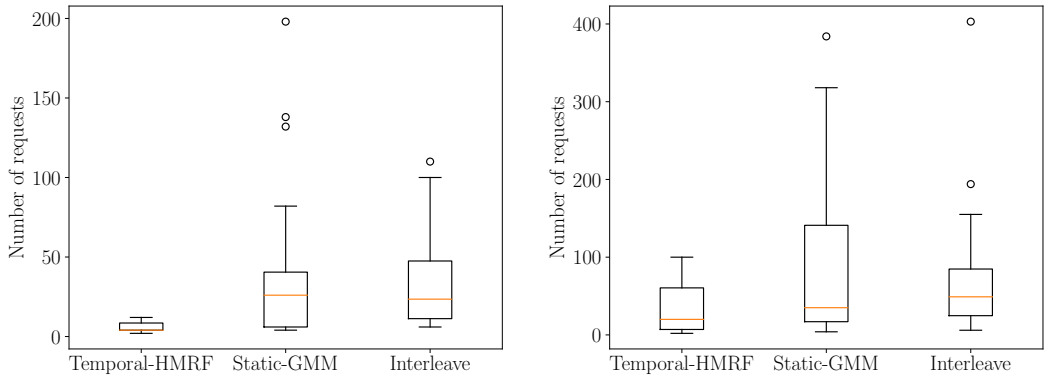


(c) B3F4



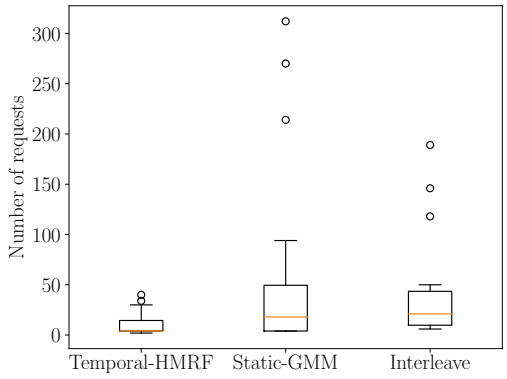
(d) B3F5

Figure A.2: Results for F2-F5 with B3 as a function of the amplification rate β.

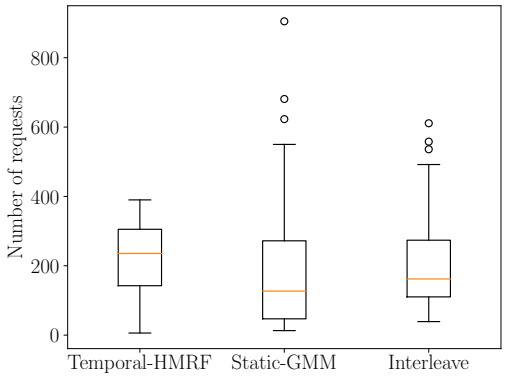


(a) B1F2 ($\beta = 0.2$)

(b) B1F3 ($\beta = 0.2$)

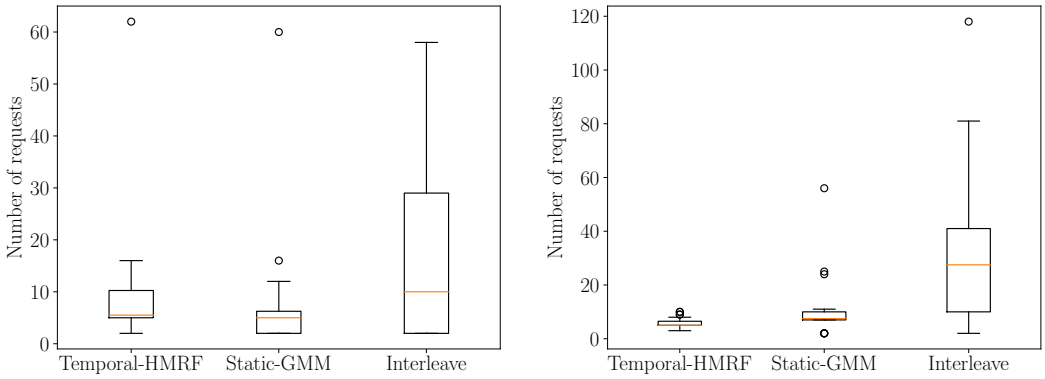


(c) B1F4 ($\beta = 0.2$)



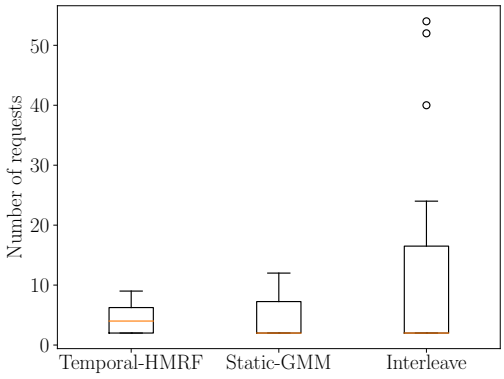
(d) B1F5 ($\beta = 0.2$)

Figure A.3: Box plots for F2-F5 with B1 ($\beta = 0.2$).

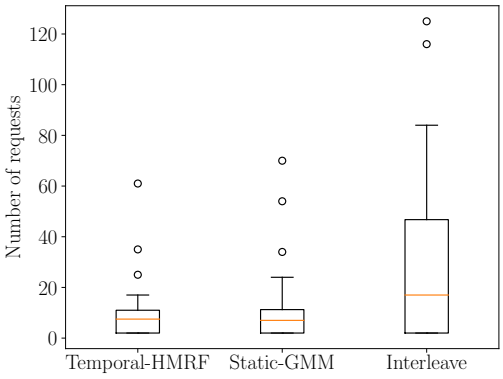


(a) B2F2 ($\beta = 0.1$)

(b) B2F3 ($\beta = 0.1$)

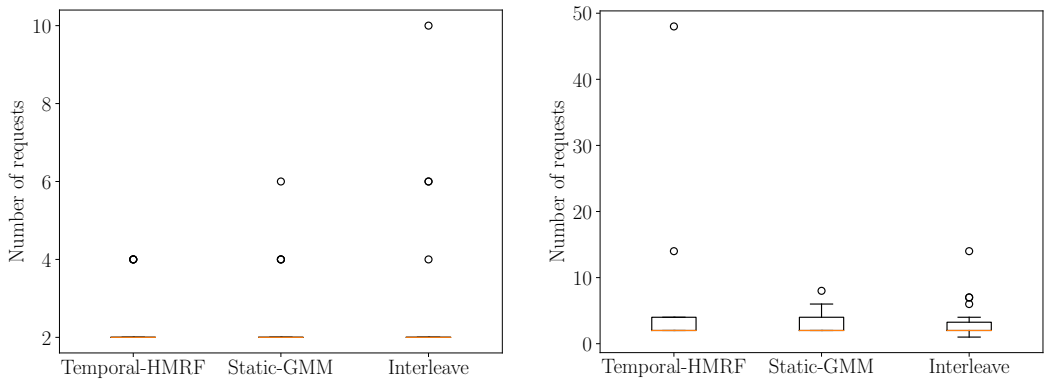


(c) B2F4 ($\beta = 0.1$)



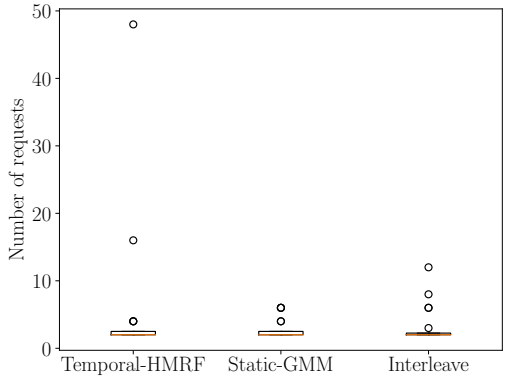
(d) B2F5 ($\beta = 0.1$)

Figure A.4: Box plots for F2-F5 with B2 ($\beta = 0.1$).

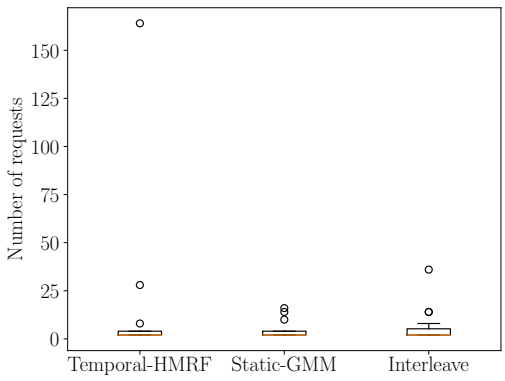


(a) B3F2 ($\beta = 0.1$)

(b) B3F3 ($\beta = 0.1$)



(c) B3F4 ($\beta = 0.1$)



(d) B3F5 ($\beta = 0.1$)

Figure A.5: Box plots for F2-F5 with B3 ($\beta = 0.1$).