THE UNIVERSITY *of* EDINBURGH

Probabilistic Models for Melodic Sequences

Athina Spiliopoulou

Doctor of Philosophy

Institute for Adaptive and Neural Computation
School of Informatics
University of Edinburgh

2013

Abstract

Structure is one of the fundamentals of music, yet the complexity arising from the vast number of possible variations of musical elements such as rhythm, melody, harmony, key, texture and form, along with their combinations, makes music modelling a particularly challenging task for machine learning.

The research presented in this thesis focuses on the problem of learning a generative model for melody directly from musical sequences belonging to the same genre. Our goal is to develop probabilistic models that can automatically capture the complex statistical dependencies evident in music without the need to incorporate significant domain-specific knowledge. At all stages we avoid making assumptions explicit to music and consider models that can can be readily applied in different music genres and can easily be adapted for other sequential data domains.

We develop the Dirichlet Variable-Length Markov Model (Dirichlet-VMM), a Bayesian formulation of the Variable-Length Markov Model (VMM), where smoothing is performed in a systematic probabilistic manner. The model is a general-purpose, dictionary-based predictor with a formal smoothing technique and is shown to perform significantly better than the standard VMM in melody modelling.

Motivated by the ability of the Restricted Boltzmann Machine (RBM) to extract high quality latent features in an unsupervised manner, we next develop the Time-Convolutional Restricted Boltzmann Machine (TC-RBM), a novel adaptation of the Convolutional RBM for modelling sequential data. We show that the TC-RBM learns descriptive musical features such as chords, octaves and typical melody movement patterns.

To deal with the non-stationarity of music, we develop the Variable-gram Topic model, which employs the Dirichlet-VMM for the parametrisation of the topic distributions. The Dirichlet-VMM models the local temporal structure, while the latent topics represent different music regimes. The model does not make any assumptions explicit to music, but it is particularly suitable in this context, as it couples the latent topic formalism with an expressive model of contextual information.

Acknowledgements

I would like to thank my supervisor Amos Storkey. Amos truly inspired me to delve into machine learning during my master's degree and taught me a huge amount throughout my PhD. I am very grateful for all his advice, encouragement and feedback and for sharing with me his scientific knowledge and insight. His contribution to this work has been invaluable. I would also like to acknowledge Chris Williams and Victor Lavrenko, whose high-level advice and feedback have helped shape the work in this thesis. In addition, I would like to thank Chris for always taking the time to read my manuscripts in depth and provide me with detailed and substantial comments.

During my PhD I was lucky to be surrounded by many talented friends and colleagues, who have greatly contributed to my learning experience. I would like to acknowledge Nicolas Heess who provided me with insightful comments and made working with RBMs so much more fun. I would like to thank Peter Orchard who helped me design and implement the basic functionality for the TC-RBM. He also continually supported and encouraged me while I was writing this thesis and for this I am grateful. I am thankful to Iain Murray for his helpful comments and advice and for always being reachable, despite his very busy schedule. I would also like to acknowledge by name Ali Eslami, Andrew Dai, Jyri Kivinen, Mark Granroth-Wilding and Mark Steedman for interesting conversations that often gave me a different perspective on my work. Finally, many thanks to John Quinn and Martin Mubangizi for a brilliant research visit at the Makerere University in Uganda. Working with them was a great experience.

I would like to express my appreciation to the School of Informatics for providing the infrastructure and funding for this work, and to Denis Howell for proof-reading and commenting on my final draft.

My deepest gratitude to my parents, Matina and Dionissis, and my sister, Fotini, for their unconditional love and for being with me always even when away. Without them this would not have been possible. Thank you also to my very good friends Adrianna Teriakidis and Laura Skylaki for many interesting distractions and for many happy moments. They made my PhD experience very enjoyable.

Finally I would like to thank Scott. For his ingenious way of handling stressful situations, for making me laugh and for giving me the motivation to finish.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Athina Spiliopoulou*)

Table of Contents

1 Introduction **1**

1.1 Machine Learning and Music 4

1.2 Probabilistic Models for Music Generation 6

2 Model Evaluation **11**

2.1 Data Likelihood 13

2.1.1 Prediction Task 13

2.1.2 Marginal Likelihood 15

2.1.3 Limitations 16

2.2 Comparing Model Samples with Data Sequences 17

2.2.1 Maximum Mean Discrepancy 17

2.2.2 The Mismatch Kernel 18

2.2.3 Limitations 19

2.3 Experimental Procedure 20

2.3.1 Dataset 21

2.3.2 Implementation Details 24

3 Bayesian Dictionary-Based Predictors **29**

3.1 Preliminaries 29

3.1.1 Markov Models 30

3.1.2 Hidden Markov Models 31

3.1.3 Variable-Length Markov Models 32

3.2 The Dirichlet Variable-Length Markov Model 35

3.2.1 Inference and Learning 37

3.2.2 Relation to other models 40

3.3 Combining Global and Local Models 42

3.3.1 Update the counts 42

3.3.2 Product Formulation 44
3.3.3 Mixing Priors 45
3.4 Model Evaluation 46
3.4.1 Analysing Meta and Hyper-Parameters 46
3.4.2 Summary of Findings 58
3.5 Comparative Analysis 59
3.5.1 Next-Step Prediction 60
3.5.2 Maximum Mean Discrepancy of String Kernels 62
3.6 Discussion 72

4 Unsupervised Latent Feature Extractors 75
4.1 The Restricted Boltzmann Machine 76
4.1.1 Inference and Learning 77
4.2 The Time-Convolutional Restricted Boltzmann Machine 79
4.2.1 Inference and Learning 84
4.3 Model Evaluation 85
4.3.1 Implementation Details 87
4.3.2 Analysing Model and Training Meta-Parameters 89
4.3.3 Learning Musical Features 103
4.4 Comparative Analysis 106
4.4.1 Next-Step Prediction 106
4.4.2 Maximum Mean Discrepancy of String Kernels 108
4.5 Discussion 117

5 Topic Models 119
5.1 Latent Dirichlet Allocation 121
5.2 The Variable-gram Topic Model 122
5.2.1 Inference and Parameter Estimation 126
5.3 Model Evaluation 128
5.3.1 Qualitative evaluation 137
5.4 Comparative Analysis 140
5.4.1 Next-Step Prediction 140
5.4.2 Maximum Mean Discrepancy of String Kernels 142
5.5 Discussion 153

6 Discussion 155

6.1 Conclusion 157

A Collapsed Gibbs Sampling in the Variable-gram Topic Model 173

A.1 Model Definition 174

A.2 Rao-Blackwellisation 175

A.3 Gibbs Sampling 177

A.4 Inference Procedure 180

Bibliography 181

List of Figures

2.1 Data representation: notes on the stave and their corresponding 1-of-26 representation. 22
2.2 Illustration of data split for cross-validation and testing. 25

3.1 An example Probabilistic Suffix Tree for a binary sequence. 34
3.2 An example Dirichlet-VMM tree for binary data. 36
3.3 Example of a degenerate sample from the GL_1-dynamic Dirichlet-VMM. 63
3.4 Barplots depicting the estimated squared MMD for different models. 67
3.5 Similarity matrices using the $(4, 1)$ mismatch kernel between testing sequences and samples from different models. 69
3.6 Truncated similarity matrices using the $(4, 1)$ mismatch kernel between testing sequences and samples from different models. . . 70
3.7 Similarity matrices using the normalised $(4, 1)$ mismatch kernel between testing sequences and samples from different models. . . 71

4.1 A Restricted Boltzmann Machine. 77
4.2 A Time-Convolutional Restricted Boltzmann Machine. 80
4.3 Average next-step prediction log-likelihood for TC-RBMs with different numbers of hidden units. 91
4.4 Average next-step prediction log-likelihood under TC-RBMs with different filter sizes. 92
4.5 Average next-step prediction log-likelihood under TC-RBMs trained with different numbers of CD steps. 93
4.6 Average next-step prediction log-likelihood under TC-RBMs trained with different mini-batch sizes. 94

4.7 Average next-step prediction log-likelihood under TC-RBMs trained with different learning rates. 95
4.8 Average next-step prediction log-likelihood under TC-RBMs trained with different learning schedules. 96
4.9 Average next-step prediction log-likelihood under TC-RBMs with different sparsity constraints. 97
4.10 Average next-step prediction log-likelihood under TC-RBMs with different structure but same number of weight parameters. 98
4.11 Average next-step prediction log-likelihood under different TC-RBMs with an equivalent number of hidden units being allowed to turn 'on'. 100
4.12 Average next-step prediction log-likelihood under different TC-RBMs plotted against the number of training epochs. 102
4.13 Weight filters for 6 hidden units of a sparse TC-RBM network trained on pieces from the key of G. 104
4.14 Two visible configurations that filter 5 responds to. 105
4.15 Barplots depicting the estimated squared MMD using the mismatch kernel for different models. 113
4.16 Barplots depicting the estimated squared MMD using the normalised mismatch kernel for different models. 114
4.17 Truncated similarity matrices using the $(4, 1)$ mismatch kernel between testing sequences and samples from different models. . . 115
4.18 Similarity matrices using the normalised $(4, 1)$ mismatch kernel between testing sequences and samples from different models. . . 116

5.1 Graphical models for Latent Dirichlet Allocation and the Variable-gram Topic Model 123
5.2 Scatter plots of latent topic allocations with respect to the music key. 138
5.3 Conditional distributions over notes for 2 latent topics given different contexts. 139
5.4 Barplots depicting the estimated squared MMD using the mismatch kernel for different models. 146
5.5 Barplots depicting the estimated squared MMD using the normalised mismatch kernel for different models. 147

5.6 Truncated similarity matrices using the $(4, 1)$ mismatch kernel between testing sequences and samples from different models. . . 150
5.7 Similarity matrices using the normalised $(4, 1)$ mismatch kernel between testing sequences and samples from different models. . . 151
5.8 Truncated similarity matrices using the $(4, 1)$ mismatch kernel between testing sequences and samples from different models. . . 152

6.1 Melody of human composed tunes represented on the stave. . . . 168
6.2 Melody of samples from different models represented on the stave. 169
6.3 Melody of samples from different models represented on the stave. 170
6.4 Melody of samples from different models represented on the stave. 171
6.5 Melody of samples from different models represented on the stave. 172

List of Tables

3.1 Average next-step prediction log-likelihood under the VMM using 10-fold cross-validation. 48
3.2 Average next-step prediction log-likelihood under the Dirichlet-VMM using 10-fold cross-validation. 50
3.3 Average next-step prediction log-likelihood under the GL_1 combination of the Dirichlet-VMM using 10-fold cross-validation. 53
3.4 Average next-step prediction log-likelihood under the GL_2 combination of the Dirichlet-VMM using 10-fold cross-validation. 54
3.5 Average next-step prediction log-likelihood under the GL_3 combination of the Dirichlet-VMM with a fixed tree structure for the local model using 10-fold cross-validation. 56
3.6 Average next-step prediction log-likelihood under the GL_3 combination of the Dirichlet-VMM with a dynamic tree structure for the local model using 10-fold cross-validation. 57
3.7 Average next step prediction log-likelihood under a local Dirichlet-VMM using 10-fold cross-validation. 58
3.8 Average next-step prediction log-likelihood of held-out data under different models. 60
3.9 Results for the estimated squared MMD between testing sequences and samples from different models using the mismatch and the normalised mismatch kernels. 64
3.10 Results for the estimated squared MMD between training sequences and samples from different models using the mismatch and the normalised mismatch kernels. 65

4.1 List of model and learning meta-parameters and the different values we examine. 88

4.2 Best and worst performing TC-RBM configurations. 90
4.3 Subset of model and learning meta-parameters we examine with respect to the training epochs. 101
4.4 The optimal meta-parameter settings for the 6 TC-RBM networks used for model comparison. 106
4.5 Average next step prediction log-likelihood of held-out data under different models. 107
4.6 Results for the estimated squared MMD between testing sequences and samples from different models using the mismatch and the normalised mismatch kernels. 109
4.7 Results for the estimated squared MMD between training sequences and samples from different models using the mismatch and the normalised mismatch kernels. 110

5.1 Average next-step prediction log-likelihood under Latent Dirichlet Allocation using 10-fold cross-validation. 132
5.2 Average next-step prediction log-likelihood under the Bigram Topic model using 10-fold cross-validation. 133
5.3 Average next-step prediction log-likelihood under the Variable-gram Topic model with a shallow tree using 10-fold cross-validation. . . 134
5.4 Average next-step prediction log-likelihood under the Variable-gram Topic model with a deep tree using 10-fold cross-validation. 135
5.5 Optimal hyperparameter settings for the topic models used in the comparative analysis. 140
5.6 Average next step prediction log-likelihood of held-out data under different models. 141
5.7 Results for the estimated squared MMD between testing sequences and samples from different models using the mismatch and the normalised mismatch kernels. 144

6.1 Average next step prediction log-likelihood of held-out data under different models. 158
6.2 Results for the estimated squared MMD between testing sequences and samples from different models using the mismatch and the normalised mismatch kernels. 161

Chapter 1

Introduction

In this work, we examine the problem of modelling music melody as a generative process that can be learned directly from a set of musical pieces belonging to the same genre. This is an interesting task for machine learning methods. In Western music, pieces are typically composed according to a system of musical organization, rendering musical structure as one of the fundamentals of music. Nevertheless, characterizing this structure is particularly difficult as, not unlike many sequential data, melody exhibits complex statistical dependencies.

The structure of melody depends on several musical elements, such as scale, rhythm and meter, and also on how these elements and other melodic parts are organised both within single time frames and across time. Therefore, although pieces in the same genre use the same structural elements, e.g. similar motifs or phrases, we have an infinite number of possible variations of melody, making each musical piece distinct from other pieces. At the same time, repetition of melodic phrases is essential in Western music and typically promotes global coherence within a piece. However, repetition in melody can occur in almost arbitrary points in time and with different degrees of variation. Distinguishing between the similarity across pieces and within pieces adds an extra level of complexity in characterising the structure of music.

In the literature, the complexity of sequential data is often dealt with by incorporating domain-specific knowledge in the models, for instance by considering application-driven feature extraction or by designing a model in a way that reflects the structure of the data. However, the purpose of this work is rather different.

Our goal is to develop a probabilistic framework that can capture the complex statistical dependencies of sequential data, without the need to hand engineer explicit, domain-specific priors or modelling formalisms. Therefore, at all stages, we avoid representations and choices that make significant use of domain knowledge and assess how much of that knowledge is automatically captured by the models.

Capturing the statistical regularities within a musical genre is a first step towards realistic music generation. Additionally, using unsupervised learning methods to identify and represent such dependencies can be beneficial for musical tasks besides music generation. In unsupervised learning the goal is to find the hidden structure in the data, i.e. to learn descriptive features of the underlying structure of music. These features summarise (part of) the data structure and can be useful input variables in a variety of musical tasks, including genre classification and music retrieval.

The contributions of this work are as follows. Firstly, we extend the Variable-Length Markov Model (VMM), which is regarded as state-of-the-art in automatic melody generation, by introducing the Dirichlet-VMM. In this novel form, inference is performed using Bayesian agents, which at each stage utilise all available information to form their beliefs. The model allows us to capture variable-order Markov dependencies, while maintaining information coming from the shorter statistics into the predictive probabilities of the longer ones.

Secondly, we propose the Time Convolutional Restricted Boltzmann Machine (TC-RBM), a novel way of using the Restricted Boltzmann Machine (RBM) to model sequential data. Unlike previous approaches, which attempt to capture the temporal relations by introducing directed connections from previous time-steps, the TC-RBM is a fully undirected network that attempts to capture the structure of music at a motif (subsequence) level rather than a single time-step. We believe that models based on the RBM are particularly suitable for capturing the componential structure of music, as they can learn distributed representations of the input space, decoupling the different factors of variation into features being 'on' or 'off'. Models based on the RBM have been shown to be good latent feature extractors in a wide range of applications.

Next, we develop the Variable-gram Topic model, which couples the latent topic formalism with a systematic model of contextual information, by employing the

Dirichlet-VMM for the parametrisation of the topic distributions over words. The latent topics allow the model to represent different music regimes and to distinguish between inter and intra-sequence structure. The model does not make any assumptions explicit to music, but it is particularly suitable in the musical context, as it is able to model temporal dependencies of considerable complexity without enforcing a stationarity assumption for the data.

Finally, we introduce a novel way of evaluating generative models for discrete data, which employs the Maximum Mean Discrepancy and string kernels to directly compare model samples with data sequences.

The rest of the thesis proceeds as follows:

- In the remaining Sections of this Chapter we give an overview of the literature on machine learning and music and review approaches to modelling musical structure, which is the problem of interest in this thesis.

- In Chapter 2 we examine the issue of evaluation in the context of melody modelling. We define the metrics that we are going to use throughout the thesis to assess model performance and we give a description of the dataset and our choices regarding data representation.

- In Chapter 3 we begin by reviewing the variable-length Markov model. We introduce the Dirichlet-VMM and describe inference and learning. We evaluate model performance using a cross-validation procedure and finally compare the performance of the Dirichlet-VMM to a standard VMM and other selected baselines, using a held-out testing set.

- In Chapter 4 we review the Restricted Boltzmann Machine and develop the Time Convolutional Restricted Boltzmann Machine. We analyse the behaviour of the TC-RBM with respect to different learning meta-parameters and then compare model performance against the Dirichlet-VMM models from Chapter 3 and other baselines.

- In Chapter 5 we begin by summarising Latent Dirichlet Allocation, one of the most commonly used topic models. We then introduce the variable-gram topic model and describe inference and learning. We analyse model performance with respect to hyper and meta-parameters and finally perform a comparative analysis using topic models and other related non-topic

models.

- Finally, in Chapter 6 we draw conclusions regarding the melody modelling problem, summarise the main contributions of this thesis, and discuss possible directions for future work.

1.1 Machine Learning and Music

Music has always been a fascinating research area for people working in fields as diverse as mathematics, the arts and philosophy. In the machine learning literature we also find applications that deal with musical tasks. These range from audio signal transcription and beat tracking to music recommendation systems. Depending on the type of data that is examined we can distinguish three overlapping categories of music applications that can be tackled using statistical machinery: applications that attempt to model aspects of the raw audio signal of music pieces; applications that attempt to model elements of musical structure using symbolic representations, such as MIDI; and applications that use meta-data, possibly in conjunction with features extracted by either of the two previous data types, in order to perform music-related tasks such as genre classification and music information retrieval.

The main application in the first category is music transcription, where the goal is to convert the audio signal to a symbolic representation. This includes several sub-problems such as pitch tracking, where the task is to detect the fundamental frequency of notes in the audio signal and beat tracking, where we effectively identify the rhythm of a piece. We refer the interested reader to Cemgil (2004) and references thereafter for a review of methods that deal with music audio transcription.

In this work we focus on the problem of modelling melody using a discrete representation of music, in the form of MIDI files. This is a symbolic representation of music and we review applications in this area in the following section. Symbolic data allows us to avoid difficulties arising due to the nature of sound and concentrate on capturing the structural elements of music, thus opening the possibility of designing models that can generate new realistic musical sequences.

So far, computer music generation has mainly been performed by expert systems.

However, the advent of the MIDI standard and the growth of MIDI databases available online have rendered probabilistic generative models appropriate for the task. In contrast with rule-based approaches, which are entirely hand-crafted, approaches using probabilistic models try to learn the musical structure directly from the data. Consequently, they are less human-labour intense and can be applied in different music genres without much additional effort. Moreover, probabilistic models can be more "creative" than deterministic ones, since they can capture the variations that we encounter in different kinds of music and generate new pieces in a stochastic manner while still respecting the rules of composition.

MIDI (Musical Instrument Digital Interface) is an industry-standard protocol for symbolic representation of music. Instead of the audio waveform, MIDI expresses music with a sequence of "event messages" that describe the basic attributes of sound. More specifically, each event corresponds to a key being struck and is described by five attributes, namely, onset time, pitch, duration, velocity and channel (instrument). Musical pieces are represented by temporal sequences of such events. Using these five attributes, MIDI files encode — either explicitly or implicitly — the four most basic elements of music, that is pitch, rhythm, loudness and timbre. Therefore, with a small loss of information, MIDI files offer a much compressed representation of music, free from inherent difficulties of sound which are present when dealing with the waveform.

Given a set of pieces from the same genre, the music components that we are typically interested in modelling are rhythm, chord progressions and melody. These elements are more or less influenced by the musical form of a piece, that is the "architectural" structure with respect to which a piece is built. The musical form is distinctive among different genres and is determined by a combination of various factors, that range from harmony and instrumentation to cultural and geographical origins. Consequently, pieces from the same musical genre are expected to exhibit complex statistical regularities among their properties. Identifying and representing these dependencies is crucial for realistic music generation, for instance in accompaniment systems and computer-aided composition systems. Additionally, this information can be used to boost the performance on a variety of musical tasks such as genre classification, recommendation systems and music retrieval, or as prior information in audio transcription systems.

1.2 Probabilistic Models for Music Generation

The most competent early approaches in modelling music are based on Recurrent Neural Networks (RNNs). Mozer (1994) proposes CONCERT for algorithmic music composition. This system selects notes sequentially, using the probability of the next note given the previous context. Predictions are based on a fully-connected RNN that receives input from the previous time-step. The network is trained on a set of pieces from the same musical style and is then used to compose new pieces through note-by-note prediction. CONCERT performs well in simple structured artificial music, but is unable to generate realistic music when trained on real music sets. According to the authors, the novel compositions are better than generations from third-order Markov models, but still lack musical coherence due to the inefficiency of RNNs in learning long-term dependencies. Moreover, CONCERT's superiority to Markov models can be partially accredited to the fact that the units of the RNN encode pitch, duration and chord at each time-step in a distributed, musically-sound manner that biases the predictions towards harmonically related notes.

In order to deal with long-term relations, Eck and Schmidhuber (2002) propose the use of Long-Short Term Memory (LSTM) RNNs. The main motivation is that LSTM networks can store memories that persist over time, by using specific representational resources, called *cells*, which offer a partial solution to the vanishing gradient problem of normal RNNs. In Eck and Schmidhuber (2002), LSTMs are shown to be successful in two, relatively simple, musical tasks; learn a specific 12-bar chord progression used in bebop jazz and learn to generate music by reproducing this chord structure and constraining the melody according to each current chord. In both tasks, the LSTMs are fully-connected and receive input from the previous time-step. In the second task, the LSTM for melody is additionally conditioned on the current chord. The compositions respect the structure of the musical form and thus assert the ability of LSTMs to learn repetitive patterns. However, the learning task is much simpler than in real-world musical datasets. The training set is generated using a single chord progression and concatenations of musically compatible melody segments that include only quarter notes and have no rests.

Franklin (2006) examines the effectiveness of different LSTM networks and dif-

ferent music representations in several short musical tasks. Representations that incorporate music knowledge — similar to Mozer's — are found to increase the performance of the networks and more complex tasks can be performed by combining networks trained on sub-tasks. For instance, the melody of a song is reproduced by combining two LSTM networks, separately trained on pitch and duration.

In Eck and Lapalme (2008) the LSTM network is extended to include information with respect to the meter. The motivation lies on the fact that although the LSTM is capable of reproducing a long sequence, it is not able to repeat smaller patterns in a different or arbitrary order. By providing information about the meter, the network is biased to learn patterns that respect the metrical structure which leads to a kind of "content-driven" memory during generation. This information is fed to the network in the form of time-delayed input which is the observation from the corresponding time-step in the preceding two, four and eight measures. The model is trained on a set of traditional Irish reels and is shown to generate pieces that respect the reel style.

A different approach to modelling polyphonic music is proposed in Lavrenko and Pickens (2003). Each MIDI file is represented as a 2-dimensional binary graph that has time and pitch along the x and the y-axis respectively. At any point in time each unit is "on" if the corresponding pitch is played in either the melody or the chord of a piece. MIDI files are transformed to this representation and used to learn a directed Random Field (RF) which is limited to connections coming from the past or the present, i.e. nodes cannot be influenced by nodes that follow. This restriction follows from the temporal nature of the data and allows for next note prediction. The advantage of using an RF is the ability to define arbitrary features that encode different dependencies we may expect the data to have. However, in order for the model to remain tractable these features need to be carefully selected and are typically restricted to neighbouring nodes in the graph. For instance, in Lavrenko and Pickens (2003), information about the octave and the duration is dropped, so that the graph is small enough for the RF to remain tractable.

In Weiland et al. (2005), pitch structure is modelled using a Hierarchical Hidden Markov Model (HHMM). This approach is motivated by the efficiency of simple HMMs in modelling local dependencies and the fact that composition using a hierarchy of simper structures can lead to music with global consistency. The twelve pitch-classes, together with two symbols for the beginning and the end of a

phrase are the emissions of the lower-level HMM; no information about duration is included. The training set consists of twenty-five Bach chorales, transposed in C-major. The HHMM has three internal states that represent "start", "body" and "end" phrases, which is a simple, yet valid way of decomposing chorales. Training is performed with the EM algorithm, but in order to learn sensible parameters, the model needs to be initialised and tuned with respect to musical knowledge. Therefore, encoding more information or modelling different genres of music with HHMMs relies on finding good representations of musical aspects, so that the architecture of the model respects particular structural aspects, while learning remains tractable.

Another extension of Hidden Markov Models that has been used for music modelling is the Input-Output HMM (IOHMM). In Paiement et al. (2009a), 53 jazz standards' melodies, manually annotated with chord labels, are used to train an IOHMM for melodic prediction. The model extends the standard HMM by further conditioning the hidden state at each time step on an observed input. The melody is the output of the model, a 13-valued categorical variable having one emission symbol for each pitch-class and one for silence. On the other hand, the input encodes the current chord. The IOHMM model is compared in terms of next-step prediction accuracy with a simple, local model that conditions the output directly on the input and assumes independently and identically distributed (i.i.d.) observations. However, the focus of the paper is to compare different chord representations rather than evaluate the quality of the model generations. Therefore, it is not clear if the IOHMM model can generate realistic music sequences.

In Paiement et al. (2009b) the former IOHMM model is compared in terms of melodic prediction accuracy with a directed graphical model, with the latter one performing better in two different datasets. The graphical model for melody is part of a larger, generative, probabilistic framework for music generation and incorporates valuable musical knowledge. In this framework, different musical sub-tasks are initially tackled separately and the information derived from the separate models is then combined for melodic prediction using the final directed graphical model, which is conditioned on everything else.

The overall framework is detailed in Paiement (2008). Chord progressions are modelled with a tree-like structure, whose parameters are tied with respect to structural aspects of a specific genre. The model is trained on chord progressions

from 52 jazz pieces using the EM algorithm and appropriate initialisation. Rhythm prediction is achieved by optimising an HMM subject to a constraint, which is derived from a musically-inspired model of distances between rhythms. Finally, melodic prediction is achieved by a directed graphical model, in which melody is conditioned on the chords and a set of features that incorporate information from the rhythm model. These are called Narmour features and are derived from a complex musicological model for musical expectation. This framework is shown to perform better than standard machine learning techniques and is therefore expected to generate music that respects local constraints and has global consistency. However, the overall framework is hand-engineered to incorporate music knowledge and is not directly applicable to other sequential datasets. Additionally, certain aspects of the models, such as the depths of the trees and the tying of the parameters, need to be tailored to the particular structure of each musical genre.

Another interesting music problem examined in the literature is harmonisation, which looks at the dual formulation of the approach described above. The task here is to find the chord progression in a music piece given the corresponding melodic sequence. In most genres, this is an easier task because chord changes are typically found at fixed time intervals, and are therefore simpler to detect compared to changes in the melody line, which can happen at almost arbitrary points in the music signal. Harmonisation of Bach chorales has been very successfully tackled in Allan and Williams (2005). They use two HMMs one for chord progressions and one for "ornamentation". In Bach chorales the chords are 4-note constructions. In the first HMM the hidden state is defined by the three notes of the chord and the observation is the fourth note. The transition probabilities between hidden states and the emission probabilities of the fourth note are learned from data. The second HMM allows the resulting 4-note chord to be modified by an ornamentation mechanism. Paiement et al. (2006) propose a tree-like graphical model for the problem of harmonisation. This still respects the structural and harmonic elements of the musical style, but it is no longer restricted to having a 4-voice structure as input.

Another successful line of research has examined the application of methodologies from the fields of statistical language modelling and text compression to the modelling of music. Dubnov et al. (2003) propose two dictionary-based predic-

tion methods, Incremental Parsing (IP) and Prediction Suffix Trees (PSTs), for modelling melodies with a Variable-Length Markov model (VMM). Despite their fairly simple nature the VMM methods are able to capture both large and small order Markov dependencies and according to Paiement (2008) achieve impressive musical generations. This is attributed to the ability of the VMM to parse music into a lexicon of phrases or motifs coupled with prediction probabilities, which can be used as a set of stochastic grammar rules that, when applied, generate new instances of music.

The VMM is further examined in Begleiter et al. (2004), who perform a comparative analysis of six different algorithms for training a VMM. The algorithms differ in the way they handle the counting of occurrences, the smoothing of unobserved events and the modelling of the variable-length and their performance is evaluated on protein classification, and sequence prediction of English text and music pieces. Finally, an interesting application of dictionary-based predictors in the music context is presented in Conklin and Witten (1995) and further examined by Pearce and Wiggins (2004). These works describe a multiple viewpoint system comprising a cross-product of Prediction by Partial Match (PPM) models. Each PPM is trained on a different representations (viewpoints) of monophonic (melodic) sequences, such as chromatic pitch, pitch contour, onset time and duration. For each of these viewpoints they construct two PPMs, a long-term (LTM), whose parameters are estimated on the entire training corpus and updated during prediction and a short-term (STM) which is constructed online for each composition in the test set.

Chapter 2

Model Evaluation

In this Chapter we examine the issue of evaluation in the context of melody modelling. We are looking at an unsupervised learning problem, where we are given a set of input sequences belonging to the same genre and we want to automatically discover statistical regularities in the data. Our ultimate goal is to learn a probability distribution that is consistent with the musical form of a genre, so that samples from this distribution would sound similar to true songs belonging to the genre. Therefore, the aim of evaluation is to assess the ability of a probabilistic model to learn the inherent structure of melodic sequences belonging to the same music genre.

This is a difficult task as we do not have a true theoretical distribution of a genre to compare our models against. Therefore, any measurement method attempting to quantify the distance from such a distribution is, at least implicitly, dependent on some modelling assumption or criterion. In the music modelling literature, evaluation is often based on qualitative analysis, like listening to model generations and analysing the structure and parameters of a model with respect to music theory and knowledge about the structure of a musical genre. To our knowledge, the quantitative measures used so far are next-step prediction accuracy (Paiement et al., 2009b; Lavrenko and Pickens, 2003), perplexity (Lavrenko and Pickens, 2003) and log-loss (Begleiter et al., 2004). These measures are typically computed by recasting the problem as a time-series prediction task. The quantity of interest is then the probability distribution over the next time-step given preceding time-steps and it is used to either make a prediction and compare to the true event or

to estimate how likely is the entire sequence under a model. Such approaches are discussed in Section 2.1.

In this work, we want to assess model attributes that go beyond their ability to make predictions. We are interested in understanding how well different types of models can automatically capture the melodic structure and whether they can generate samples that are locally or globally or both locally and globally consistent with a musical form. In Section 2.2 we describe a novel quantitative evaluation framework that attempts to estimate how close a model distribution is to the true theoretical data distribution by comparing artificially synthesised sequences — these are sampled from a model — to true data sequences — these are songs from the same genre that a model has not seen before.

Alternatively, we could assess model performance through human evaluations. For instance, we could play a sample from each model to experienced listeners and ask them to choose the one that resembles the musical genre the most, or ask them to rate the samples from the most to the least musically sounding one. Human evaluations have often been used in tasks from the music information retrieval (MIR) field. These are typically performed through the Music Information Retrieval Evaluation eXchange (MIREX) (Downie, 2008), a community-based evaluation framework for systems and algorithms used in MIR research. For example, the Evalutron 6000 (Gruzd et al., 2007) was used to evaluate systems for "Audio Music Retrieval" and "Symbolic Music Retrieval", where human evaluators had to score the audio or symbolic similarity between a query and a candidate answer, respectively.

Another example of human evaluation for a music task is presented by Barrington et al. (2009). They examine music recommendation systems and describe a new platform, that can be used to evaluate playlists by humans. They explain the experimental design, i.e. the navigation of the subjects through the experiment website, when they listen to songs, how many, how is the evaluation performed, and so on, and the factors according to which a playlist is considered a good or a bad match for the given song query. Although a human evaluation task could be well-suited for measuring the ability of models to synthesise realistic music sequences, establishing such an evaluation paradigm can be difficult and is outside the scope of this work. Therefore, for the purpose of evaluation, we mainly rely on the two quantitative measures which are described in this Chapter.

The Chapter is structured as follows. In Sections 2.1 and 2.2 we give descriptions of the quantitative methods we use to evaluate the performance of different models in this learning setting. The first one is next-step prediction log-likelihood, which effectively measures the probability of held-out sequences under a model. This is a well-established metric for evaluating performance in probabilistic models and shows how well a model can generalise to data it has not seen during learning. The second one attempts to quantify the resemblance of sequences sampled from a model to true data sequences, by employing string kernels and the Maximum Mean Discrepancy (Gretton et al., 2007). Finally, in Section 2.3 we describe the collection and preprocessing of the data and give specific details on how we implement the evaluation methods.

2.1 Data Likelihood

In pattern recognition and machine learning, model selection is often performed by assessing the ability of the model to generalise on new data, that is by measuring how well the model can perform the task at hand on data that it has not seen during learning. In the unsupervised learning setting examined here, where we want to determine the distribution from which the data comes from, a straightforward performance metric is the probability or likelihood of a held-out dataset under a model.

Let $\mathbf{X} = \left\{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^D\right\}$ be a dataset comprising D data vectors. The likelihood of the data $\mathbf{X}$ under a model $\mathcal{M}$ with a set of adjustable parameters $\boldsymbol{\theta}$ is given by

$$p(\mathbf{X}|\boldsymbol{\theta}, \mathcal{M}) = \prod_{d=1}^{D} P_{\mathcal{M}}(\mathbf{x}^d|\boldsymbol{\theta}) \ , \tag{2.1}$$

where $P_{\mathcal{M}}(\cdot|\boldsymbol{\theta})$ denotes the probability distribution defined by model $\mathcal{M}$ with parameters $\boldsymbol{\theta}$ and we have assumed that the data vectors are independently and identically distributed according to $P_{\mathcal{M}}(\cdot|\boldsymbol{\theta})$.

2.1.1 Prediction Task

In the case of melody modelling, each data vector is a sequence of observations through time, with $\mathbf{x}^d = \left\{x_1^d, \ldots, x_t^d, \ldots, x_{T_d}^d\right\}$. Using the product rule of prob-

ability we can re-write the likelihood function in (2.1) in terms of a time-series next-step prediction task as follows

$$p(\mathbf{X}|\boldsymbol{\theta}, \mathcal{M}) = \prod_{d=1}^{D} P_{\mathcal{M}}(x_1^d|\boldsymbol{\theta}) \prod_{t=2}^{T_d} P_{\mathcal{M}}(x_t^d|x_{t-1}^d, \ldots, x_1^d, \boldsymbol{\theta}) \ . \tag{2.2}$$

The first metric that we use to evaluate model performance is the average next-step prediction log-likelihood of held-out data sequences under each model. Keeping the notation from above, for a dataset $\mathbf{X}$ and a model $\mathcal{M}$ with parameters $\boldsymbol{\theta}$ this is defined as

$$\mathcal{L}(\boldsymbol{\theta}, \mathcal{M}; \mathbf{X}) = \frac{1}{\sum_{d=1}^{D} T_d} \sum_{d=1}^{D} \sum_{t=1}^{T_d} \log P_{\mathcal{M}}(x_t^d|x_{t-1}^d, \ldots, x_0^d, \boldsymbol{\theta}) \ , \tag{2.3}$$

where x_0^d is a null observation specifying the absence of events, the logarithm is taken to base e and T_d is the length of sequence d, so that the denominator is the total number of time-steps in $\mathbf{X}$. It is easy to see that this is the logarithm of the data likelihood in (2.2), averaged across all time-steps from the data sequences.

Therefore this evaluation metric shows the average log probability that a model assigns to each time-step. Note that the logarithm is a monotonically increasing function and thus maximising the next-step prediction log-likelihood in (2.3) is equivalent to maximising the likelihood of the data under model $\mathcal{M}$ with parameters $\boldsymbol{\theta}$.

The negative of the data log-likelihood is often called the log-loss, which has the following interpretation in the context of information theory. Let $P_{\mathcal{M}}$ denote the distribution of model $\mathcal{M}$ and $P_{\mathcal{T}}$ denote the true theoretical distribution that generated the data. The cross-entropy between these two distributions is defined as

$$H(P_{\mathcal{T}}, P_{\mathcal{M}}) = H(P_{\mathcal{T}}) + D_{KL}(P_{\mathcal{T}}||P_{\mathcal{M}}) = -E_{P_{\mathcal{T}}}\left[\log P_{\mathcal{M}}\right] \ , \tag{2.4}$$

where D_{KL} denotes the Kullback-Leibler divergence and $E_P\left[\cdot\right]$ denotes an expectation with respect to distribution P. If the logarithm is taken to base e, the cross-entropy measures the average number of extra nats needed to compress an event if we use coding scheme $P_{\mathcal{M}}$ instead of the true source that generated the data $P_{\mathcal{T}}$ and is uniquely minimised when $P_{\mathcal{M}} = P_{\mathcal{T}}$.

If we consider a dataset $\mathbf{X} = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^D\}$, with $\mathbf{x}^d = \{x_1^d, \ldots, x_{T_d}^d\}$ and an empirical distribution $P_{\mathcal{E}}$, which assigns probability $1/\sum_d T_d$ to every data point, then this empirical version of the cross-entropy is given by

$$H(P_{\mathcal{E}}, P_{\mathcal{M}}) = -\frac{1}{\sum_d T_d} \sum_{d=1}^{D} \log P_{\mathcal{M}}(\mathbf{x}^d) = -\mathcal{L}(\boldsymbol{\theta}, \mathcal{M}; \mathbf{X}) \ , \tag{2.5}$$

which is the negative of the average next-step prediction log-likelihood defined in (2.3).

2.1.2 Marginal Likelihood

The need for a held-out data set arises from the fact that a model with many adjustable parameters can improve its performance on the training set by fitting the noise in the data, where by noise we mean specific details that may arise in a finite data sample but are not part of the structure that characterises the general population we wish to model. This undesirable property is known as overfitting and is common when the number of adjustable parameters is large or when the supply of data is limited.

In Bayesian model selection the problem of overfitting is alleviated by considering the marginal likelihood, often referred to as the data evidence. To do this we introduce a prior distribution over the model parameters, which represents our uncertainty about their value. The marginal likelihood is a data-dependent function that shows how likely it is that the data was generated from a specific type of model, having marginalised over all possible parameter values. Following the notation from above, we have

$$p(\mathbf{X}|\mathcal{M}) = \int p(\mathbf{X}|\boldsymbol{\theta}, \mathcal{M}) P(\boldsymbol{\theta}|\mathcal{M}) d\boldsymbol{\theta} \ . \tag{2.6}$$

It is easy to see that the marginal likelihood does not depend on the adjustable model parameters and thus can be used to directly compare the suitability of different types of models for a given problem. However, in models that involve non-conjugate priors, computing the marginal likelihood can be computationally expensive or intractable.

For this reason, in the following sections we resort to using held-out datasets to perform both model selection and model comparison. More specifically, for each

type of model, we first select an optimal parameter setting, $\boldsymbol{\theta}^*$, through a 10-fold cross-validation procedure and then we compare the optimal setting for different types of models using a held-out testing set. This procedure is detailed in Section 2.3.2.

2.1.3 Limitations

Although the data log-likelihood is a well-studied and indicative metric of model performance, it cannot always depict or fully represent specific model attributes that we may be interested in. For instance, in the topic modelling literature, Chang et al. (2009) show that if we care about the semantic meaning of a model's latent topic space, then held-out log-likelihood can lead to choosing suboptimal models with respect to this criterion.

In melody modelling we effectively try to determine the distribution from which the data is generated. This is an unsupervised learning problem involving data with complex statistical dependencies. In this setting, one aspect that is not fully characterised by log-likelihood is model underfitting. More specifically, log-likelihood decreases sharply if a model is overfitting, but it does not penalise that much a model that assigns lots of its probability mass to improbable configurations.

This means that to maximise the data log-likelihood, a model has to capture any possible melodic sequence that can occur, so that it has good generalisation properties. However, in order to achieve this, a model may also assign probability mass to other configurations which are improbable under the true source of the data. This can be seen as underfitting and is not well-captured by log-likelihood. For example, using the log-likelihood, the penalty associated with giving 0 probability to a probable event is infinite, whereas the penalty associated with assigning 0.1 probability to an event that never occurs is $\log(0.9) = -0.1054$ per datapoint. Furthermore, in time-series models underfitting can have downstream implications. A single "error" during prediction can lead to an improbable area of the search space, associated with few or no datapoints, thus leading to consecutive configurations that are not supported by the data.

2.2 Comparing Model Samples with Data Sequences

In this section, we present a novel framework for evaluating the generative properties of a model, which attempts to quantify how close is a model distribution, $P_{\mathcal{M}}$, to the true theoretical data distribution, $P_{\mathcal{T}}$, based on finite samples drawn i.i.d. (independent and identically distributed) from each. More specifically, a probabilistic model for melodic sequences specifies a joint probability distribution over observed sequences. Given such a probability distribution we sample artificially synthesised sequences, $\mathbf{x}^s$, from a model, using $\mathbf{x}^s \sim P_{\mathcal{M}}(\mathbf{x})$. Additionally, we assume that true songs belonging to a genre are drawn from a theoretical distribution characterising the genre.

Given these two populations — model samples and true data — we first employ the mismatch kernel to compute a similarity score between every pair of sequences. Then we use the Maximum Mean Discrepancy (Gretton et al., 2007) to estimate the distance between the two distributions by effectively comparing the intra-population similarity scores to the inter-population ones. A small distance indicates that a model generates many of the different substructures that occur in the data. By measuring the resemblance between model samples and data sequences, this method provides a complementary quantification of model performance. Additionally, by visualising the similarity patterns within and between the two populations we can better understand the properties of the musical structure and identify elements that a model fails to capture.

We describe the methodology for this evaluation in Sections 2.2.1-2.2.3 and give details on implementation choices in Section 2.3.2.

2.2.1 Maximum Mean Discrepancy

The Maximum Mean Discrepancy (MMD) is a distance metric between probability distribution embeddings on a Reproducing Kernel Hilbert Space (RKHS). Given a set of observed data sequences $\mathbf{X} := \left\{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^D\right\}$, independently and identically distributed according to $P_{\mathcal{T}}$ and a set of sampled sequences $\mathbf{X}' := \left\{\mathbf{x}^{1'}, \mathbf{x}^{2'}, \ldots, \mathbf{x}^{D'}\right\}$ drawn i.i.d. from $P_{\mathcal{M}}$, Gretton et al. (2012) prove that an

unbiased empirical estimate of the squared population MMD can be computed as

$$MMD_u^2\left[\mathcal{F}, \mathbf{X}, \mathbf{X}'\right] = \frac{1}{D(D\text{-}1)} \sum_{i=1}^{D} \sum_{j \neq i}^{D} K(\mathbf{x}^i, \mathbf{x}^j) + \frac{1}{D'(D'\text{-}1)} \sum_{i'=1}^{D'} \sum_{j' \neq i'}^{D'} K(\mathbf{x}^{i'}, \mathbf{x}^{j'}) - \frac{2}{DD'} \sum_{i=1}^{D} \sum_{j'=1}^{D'} K(\mathbf{x}^i, \mathbf{x}^{j'}), \tag{2.7}$$

where $\mathcal{F}$ is the unit ball in a RKHS $\mathcal{H}$ and $K(\mathbf{z}^i, \mathbf{z}^j) = \langle \phi(\mathbf{z}^i), \phi(\mathbf{z}^j) \rangle_{\mathcal{F}}$ is a positive definite kernel defined as the inner product between feature mappings $\phi(\mathbf{z}) \in \mathcal{F}$.

Our choice of the MMD for the distance between probability measures is motivated by the work of Sriperumbudur et al. (2009) who show that the estimator for the MMD is strongly consistent and has good convergence behaviour, while being easy to compute. From (2.7) it is easy to see that the MMD estimator has a closed form solution that does not depend on the dimensionality of the input space. This is unlike the more well-known ϕ-divergences, e.g. the Kullback-Leibler divergence, which can be difficult to estimate in high dimensions and can exhibit arbitrarily slow rates of convergence depending on the distributions. Note that the dimensionality corresponds to the length of a music sequence (~ 256 time-steps). In earlier work (Spiliopoulou and Storkey, 2011) we considered comparing short-order model statistics with data statistics, by computing the KL divergence between the normalised frequencies of statistics computed using true data and model samples. However, we found this computation unreliable for orders higher than 3 time-steps, as the KL-divergence between statistics from model samples and held-out data sequences was in many cases lower than the KL-divergence between statistics from two different partitions of the data.

2.2.2 The Mismatch Kernel

The MMD is dependent only on the kernel defined on the unit ball in a RKHS. Therefore we can use the kernel trick to compare pairs of melodic sequences. String kernels naturally lend themselves to this problem, as they define a measure of similarity between discrete structures by comparing the set of common substrings between two sequences.

We use the mismatch kernel (Leslie et al., 2004), $K_{(k,m)}(\mathbf{x}, \mathbf{x}')$, which for a pair of sequences $\mathbf{x}$ and $\mathbf{x}'$ computes the shared occurrences of k-length subsequences that

have at most m mismatches. This kernel has been successfully used for biological sequence classification (Leslie et al., 2004) and NLP tasks (Teo and Vishwanathan, 2006).

Additionally, we report results using the normalised mismatch kernels, given by

$$\tilde{K}_{(k,m)}(\mathbf{x}, \mathbf{x}') = \frac{K_{(k,m)}(\mathbf{x}, \mathbf{x}')}{\sqrt{K_{(k,m)}(\mathbf{x}, \mathbf{x})}\sqrt{K_{(k,m)}(\mathbf{x}', \mathbf{x}')}} \ , \tag{2.8}$$

as normalisation prevents the kernel being dominated by a few very long or highly repetitive sequences.

2.2.3 Limitations

The main limitation of this method is that the estimate of the MMD is dependent on the kernel. This could also be seen as a strength, as it allows us to compare distributions defined on structured domains, like strings and graphs. However, our choice of the kernel effectively determines how we deal with higher order statistics of the probability distributions that we are interested in. Since we compare distribution embeddings in the RKHS induced by the kernel, our choice of the kernel dictates whether differences in the embeddings correspond to differences of interest in the distributions. Characterising whether specific kernels, or kernel families induce sufficiently rich RKHSs to detect differences of interest between distributions and understanding the properties of distributions resulting in small or large distances in the embedding space are active areas of research (see for example Sriperumbudur et al. (2010)).

The mismatch kernel quantifies similarity with respect to k-length subsequences and in this work we primarily use $k = 4$, which in our representation (Section 2.3.1) corresponds to half a musical bar. Therefore, it can only evaluate local structure in musical generations, rather than global coherence of synthesised sequences. For the purposes of this work, this evaluation is informative as the methods we examine target low-level sequence structure, such as motifs and their transformations, rather than large-scale musical structure, such as phrasing. To evaluate the latter we would need to consider kernels or representations that capture global properties of each sequence, such as the number of phrases within a piece, the duration of each phrase, whether phrases are repeated and in what order, and so on.

Another potential limitation of this evaluation method is that it cannot directly assess whether a model is overfitting. For instance, consider a model that simply replicates the training data. This model has not learned anything about the underlying data structure, however, the squared MMD estimated from held-out sequences and samples from this model should be very low, as the training and the held-out data come from the same distribution. To account for this limitation, in the following sections we estimate the squared MMD between model samples and data sequences using a) held-out testing sequences and b) training sequences.

Finally, despite the limitations of the method presented here, we believe that evaluating models using sampled generations is an interesting direction for future research. This method is inspired from the machine translation literature, where a system is often evaluated by comparing the machine translation output to reference human translations (see for example Turian et al. (2003) and references therein). However, in this context we do not have a one-to-one mapping between machine and human output, but instead we want to assess the properties of the two populations by looking at finite samples drawn i.i.d. from each. More specifically, when we consider a new synthesised piece of music (machine output), we no longer have a target piece (human output) to compare against. As a substitute, we evaluate a model by comparing *population* features, where the populations are true music pieces and synthetic music pieces. Therefore, we can assess whether a model has captured part of the data structure, even though we cannot draw conclusions about the quality of each individual piece. The type of structure we examine, i.e. the population features, depends on our choice of the kernel.

2.3 Experimental Procedure

In the following sections we describe in more detail the experimental procedure used to train and evaluate the models presented in this thesis. We begin with a description of the data collection and preprocessing steps.

2.3.1 Dataset

The data comprises 264 reels collected from the Nottingham Folk Music Database. Reels are traditional Scottish and Irish tunes used to accompany dances. We collected the dataset in two steps. In 2009 we downloaded all available reels in 4/4 meter written in the G-major key and in 2010 we downloaded all available reels in 4/4 meter written in the D-major key. From these tunes we only use ones for which both the MIDI file and a gif file with the corresponding musical score were available at the time of the download.

The final dataset comprises 118 tunes in the G-major key and 146 in the D-major key. All tunes have 4/4 meter and each tune is represented by its musical score and a MIDI file. Note that the downloaded MIDI files were originally derived from the score, which means that there are no elements of human performance in the data. The final dataset is available for download from the author's webpage.

We read the MIDI files into MATLAB using the MIDI toolbox (Eerola and Toiviainen, 2004). Each MIDI file is represented by a table with 7 columns: [onset-in-beats, duration-in-beats, MIDI-channel, MIDI-pitch, velocity, onset-in-seconds, duration-in-seconds], with rows representing note events through time. The components we wish to model are pitch and duration of the notes in the melody, so for each piece we only consider rows that belong to the melody channel.

2.3.1.1 Representation

The MIDI standard maps each fundamental frequency to a linear pitch space in which octaves have size 12, i.e. the distance between any two adjacent semitones is 1, and middle C ($C4$) is assigned to 60. According to this mapping, pitch can take on integer values from 1 to 120. In a melody line we would expect to observe notes found in a piano. A piano typically contains 88 keys, starting at $A0$ (assigned to 21 in MIDI) up to $C8$ (assigned to 108 in MIDI). There are several ways of encoding the pitch.

- The simplest representation is to use one categorical variable that takes on 89 different values — 88 for all possible semitones from $A0$ to $C8$ and 1 for musical rest. Any information about harmonics of the fundamental pitches is ignored, e.g. $C1$ and $C2$ are no more similar than $C1$ and $G4$.

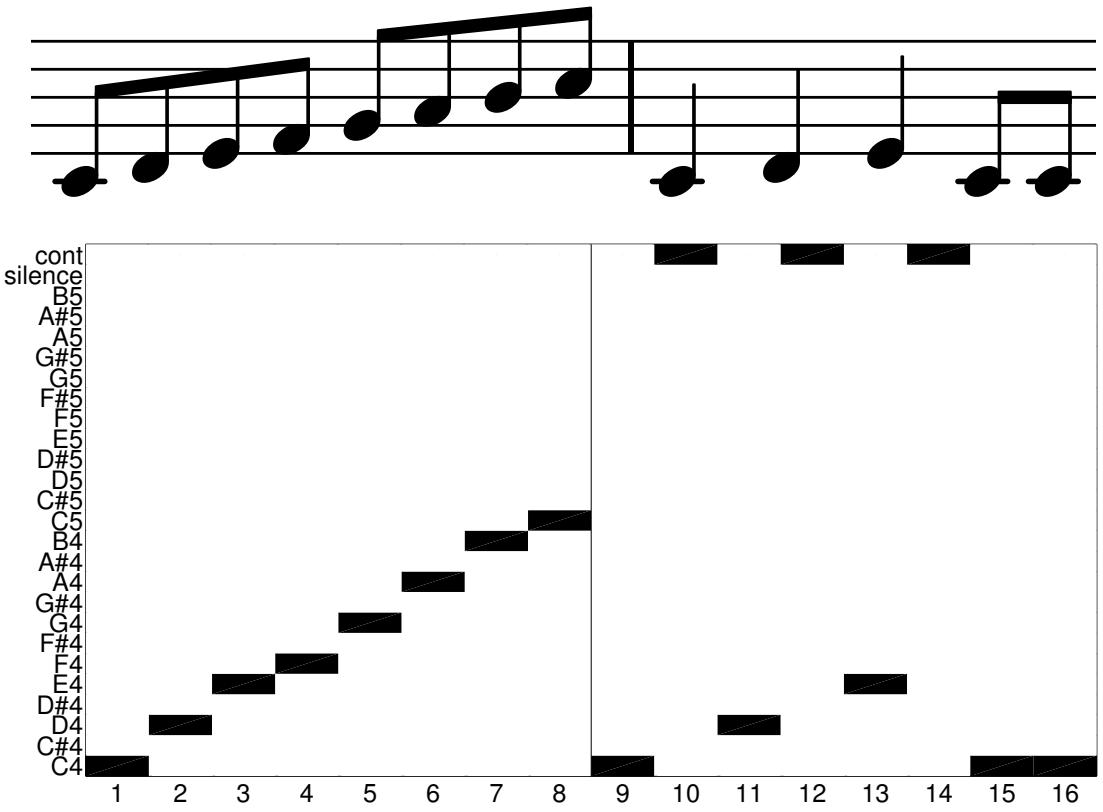


Figure 2.1: Data representation: notes on the stave and their corresponding 1-of-26 representation. Time is discretised in eighth note intervals. At each time-step, pitch is represented by a 1-of-26 categorical vector. The $C4$ quarter note (time-steps 9 and 10) is represented by $C4$ followed by continuation. This is different from the representation of two consecutive $C4$ eighth notes (time-steps 15 and 16).

- To include similarities due to harmonics, we could use 2 categorical variables, one for pitch class, i.e. the relative position within the octave, and one for pitch height, i.e. the octave. In this case the first variable can take on 13 different values ($12 + 1$ for rest), while the second one 9. Alternatively, we could have 12 variables, one for each pitch class, with each variable taking a value proportional to the absolute frequency (pitch height).

- Alternative representations could further exploit psychoacoustic similarities among the 12 pitch classes. For example, we could encode the musical relationships with respect to the circle of fifths by the (x, y) coordinate on the circle of fifths. As above, we would need to introduce a further variable to represent pitch height.

- Another choice is to encode differences between pitches instead of absolute pitch values. To represent a tune we need the absolute pitch for the first note and the sequence of successive differences between pitches. An advantage of this representation compared to absolute pitch value is that the number of possible leaps in music melody is small, resulting in a more succinct representation. Additionally, this encoding is invariant with respect to the key, i.e. if a tune is transposed to a different key, its encoding remains unchanged if we are using differences, but is completely different if we are using absolute pitch values.

Since we want to avoid incorporating domain-specific knowledge in our models we decided to use the first option to encode pitch. However, we note that the last encoding would also be a sensible choice and is not specific to the music domain. In our corpus 99.6% of the total observations fall inside the $C4$-$B5$ range. Therefore, our final representation for pitch is a 25-valued categorical variable, with one value for each possible semitone within these two octaves and one value for musical rest. The 0.4% of values outside this interval are truncated to the corresponding pitch in the nearest octave.

Additionally, we do not transpose the pieces to the same key, because we want to assess whether the models can identify this kind of structure. The G-major scale is $G\,A\,B\,C\,D\,E\,F\#\,G$, while the D-major is $D\,E\,F\#\,G\,A\,B\,C\#\,D$. Therefore, in terms of absolute pitch values, the two scales differ only in C and $C\#$. However, the function of the rest of the notes is different between scales and we want to see whether models are able to pick up this difference. We will see for instance in Chapter 5 that the topic models tend to allocate different topics in pieces from each key.

In the MIDI files duration is given in both beats and seconds. In the following we use information about beats, since this is the basic time unit in music and is robust with respect to the tempo. Duration can be modelled either implicitly or explicitly as follows

- Discretise time using a fixed time-step. In this case we only need to encode information about the pitch. However, the fraction of time that will be used must be small enough in order to account for all, or most of, the valid note durations. This results in longer notes lasting for a number of consecutive

time frames. Additionally, this encoding results in a quarter note of certain pitch being exactly the same as two consecutive eighth notes of the same pitch.

- Discretise time according to note events and use one categorical variable that can take on values for all valid note durations.
- Discretise time according to note events and use different categorical variables to encode for different time lengths. For instance, use one variable to encode how many quarter note fractions correspond to a certain duration (e.g. a dotted half note corresponds to 3 quarter note fractions), another variable to encode for sixteenth note fractions, a third one for triplets, and so on.

Here we use the first option and discretise time in eighth-note intervals. Roughly 1% of the durations cannot be represented by an eight-note interval. These events are concentrated in a few pieces, and include smaller durations, i.e. sixteenth and thirty-second notes, and triplets. The combinations that occur in the data and how they are dealt with are detailed below:

- 2 sixteenth notes $\rightarrow$ set first one to an eighth, remove the second.
- 4 thirty-second notes $\rightarrow$ set first one to an eighth, remove the rest.
- 3 eighth triplets $\rightarrow$ keep the first one as a quarter note, remove the rest.
- 3 sixteenth triplets $\rightarrow$ keep the first one as an eighth note, remove the rest.
- 1 sixteenth + 1 dotted eighth $\rightarrow$ set them to 2 eighth notes.

Finally, we augment the categorical variable used for pitch with an extra value that represents *continuation* of an event and allows us to keep more accurate information concerning the duration of notes. Our final representation is depicted in Figure 2.1. Note that a $C4$ quarter note (time-steps 9 and 10) is represented by $C4$ followed by continuation, which allows us to distinguish the case where we have two consecutive $C4$ eighth notes (time-steps 15 and 16).

2.3.2 Implementation Details

To compare different models for melodic sequences, we measure model performance on held-out datasets. There are two optimisation stages that we need to consider.

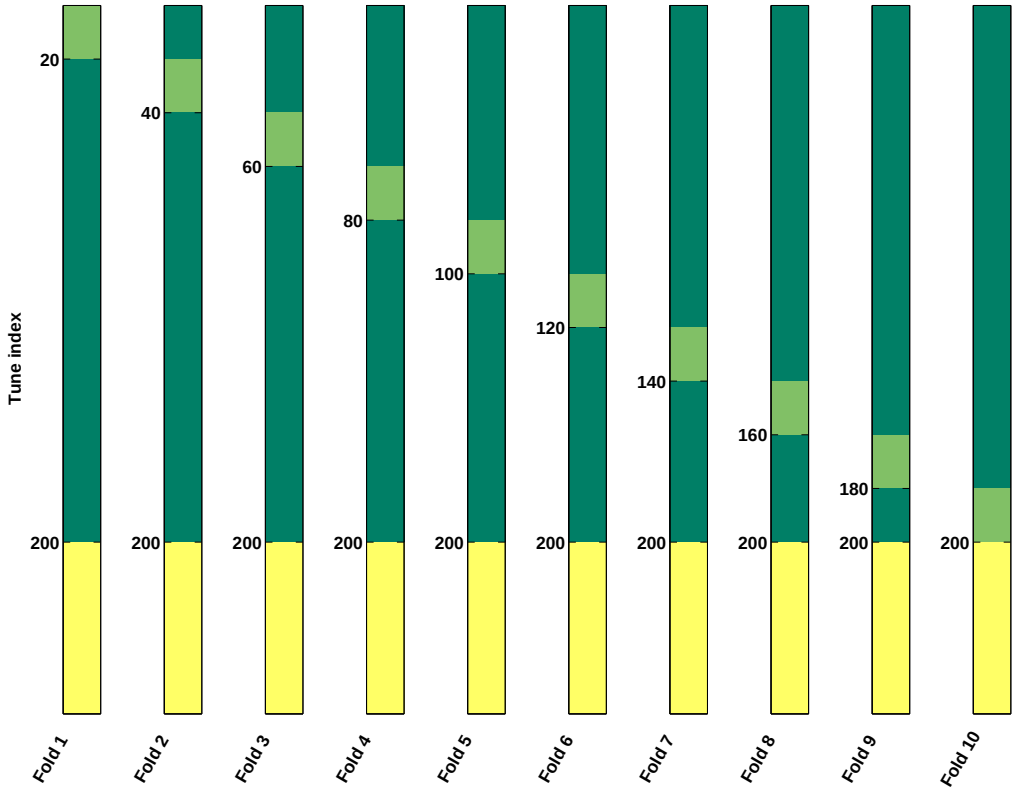


Figure 2.2: Illustration of data split for cross-validation and testing. Dark green: training data, light green: validation data, yellow: testing data. The testing data are used for model comparison, after the optimal parameter setting has been selected through cross-validation.

The first one is the actual learning problem: given a type of model, we want to learn the model parameters that give the best fit to the training data. This is typically done by optimising a loss function with respect to the training data. The second one is the optimisation of the hyper-parameters: to train a model, we typically have to choose values for several settings of the learning algorithm, or the size and structure of the model. Examples of hyper-parameters include the learning rate in neural networks, the number of experts in a mixture of experts model, and the regularisation penalty in Support Vector Machines.

In this work, to select hyper-parameters we perform a grid search over possible settings and choose the setting that optimises performance measured by cross-validation. Grid search is an exhaustive search through a prespecified subset of values from the hyper-parameter space, i.e. we specify a set of values for each hyper-parameter and subsequently train the model using every possible

combination of hyper-parameter values. The advantage of using grid search over more advanced sequential hyper-parameter optimisation algorithms or global optimisation algorithms is that it is simple to implement, as there is no technical overhead and straightforward to parallelise, as the settings are predefined.

However, grid search becomes very cumbersome and inefficient when we have many hyper-parameters, and other alternatives should be considered. In grid search the number of settings we need to examine grows exponentially with the number of hyper-parameters. However, the number of possible values for each hyper-parameter is fixed, and the same value is considered multiple times for different settings of the other hyper-parameters. Typically, only a subset of the joint hyper-parameters turns out to be important for a given dataset, and therefore many of the settings we examine during grid search become irrelevant to the optimisation problem.

Bergstra and Bengio (2012) propose random trials as an alternative to grid search trials, and show that the former is more efficient when we consider more than one hyper-parameters, while remaining easy to implement and parallelise. In random trials, each hyper-parameter setting is chosen by uniformly sampling from the configuration space that would be spanned by a regular grid. Hence for a fixed computational burden, random search examines a larger number of different values for each hyper-parameter compared to grid search, typically resulting in better exploration of the important subspace for a given dataset. Adaptive search algorithms, such as Bayesian optimisation methods which can weight the importance of each hyper-parameter in a principled manner, will result in more efficient search over the hyper-parameters, however, they come with an increased technical and implementational complexity. Bergstra and Bengio (2012) conclude that random search is generally superior to grid search for optimising hyper-parameters, and a good baseline for more sophisticated hyper-parameter optimisation methods, based on either global optimisation or adaptive search algorithms.

Figure 2.2 shows how the data is split for the purpose of evaluation. For each type of model we perform 10-fold cross-validation using 200 tunes and select the hyper-parameter setting that maximises the average next-step prediction log-likelihood in (2.3), with the average taken across the validation data from all 10 folds. To compare different types of models we use the optimal parameter setting

and compute average prediction log-likelihood on a held-out testing set comprising the remaining 64 melodic sequences. An alternative experimental design would be to use nested cross-validation. In this case, we would cross-validate both the validation step — where we select model parameters — and the testing step — where we assess model performance given a single (optimal) parameter setting. To perform nested cross-validation we need to first split the data into N_{test} folds and subsequently split the training data from each of these folds into $N_{validation}$ folds. This procedure is computationally more expensive, but less biased than the single training/testing split that we use here.

For the MMD evaluation, we want to sample melodic sequences from each model and compare them to held-out data sequences. For each model type, we use the hyper-parameter setting that maximises the log-likelihood during cross-validation in the first experiment above. Ideally we would want to select the hyper-parameter setting that optimises the MMD evaluation objective during cross-validation. However, this is computationally expensive, so we use the log-likelihood evaluation as a proxy of performance for the MMD evaluation. Having selected the model with the optimal hyper-parameter setting, we sample 64 synthesised sequences of length corresponding to the 64 held-out testing sequences. The squared MMD is then computed using the mismatch kernel between a) the sampled sequences and the held-out testing sequences and b) the sampled sequences and 64 randomly selected training sequences. The procedure is repeated in every fold and the mean and standard deviation across folds is reported.

For our computation of the mismatch kernel, we map the melodic sequences to a 25-valued categorical vector, where continuation is substituted by the corresponding pitch. This mapping removes spurious shared occurrences of continuation, where the actual pitch being continued is different. Furthermore, both the log-likelihood and the MMD evaluation are performed using only the first half of the held-out sequences, as in this genre the second half is typically exact repetition of the first half.

Finally, we use the empirical marginal distribution as a baseline for model comparison. Under the empirical marginal distribution the probability assigned to each of the 26 values is proportional to the number of times that value has been observed on the training data. This is the maximum likelihood estimate for a categorical distribution under the assumption that all time-steps are i.i.d. and thus can be

thought of as the best predictor in the absence of temporal dependencies.

Chapter 3

Bayesian Dictionary-Based Predictors

In many applications involving analysis of sequential data, the task at hand can be formulated as a prediction problem along the succession (ordering) domain. For instance, if we are dealing with temporal sequences a lot of the dependence structure can typically be captured by learning a prediction model along the time domain. Our objective then is to predict the $t + 1^{\text{st}}$ element of a sequence $\{x_1, \ldots, x_t, \ldots, x_T\}$ given that we have only observed $\{x_1, \ldots, x_t\}$.

In the following sections we examine models that attempt to capture the dependence structure in sequential data by considering a time-series prediction objective. We are looking at the case of discrete sequential data defined over a finite alphabet, Σ, and assume that the data is generated from a stationary process, i.e. the data evolves through time, but the distribution from which they come remains the same.

3.1 Preliminaries

The basic property of sequential data that requires special treatment is that the observed datapoints are not independently and identically distributed (i.i.d.) but instead they exhibit statistical regularities that depend on the ordering of datapoints. Therefore, in order to model this dependence structure in a sequence of observations $\mathbf{x} = \{x_1, x_2, \ldots, x_T\}, x_t \in \{1, \ldots, \|\Sigma\|\}$, we need to consider the

joint probability distribution $P(\mathbf{x})$.

Using the chain rule of probability we know that the joint distribution $P(\mathbf{x})$ can always be factorised as

$$P(\mathbf{x}) = P(x_1) \prod_{t=2}^{T} P(x_t | x_{t-1}, \ldots, x_1) \ . \tag{3.1}$$

This factorisation cannot be used in practice, as estimating conditional distributions that depend on all the details of previous values is computationally intensive and — for a finite amount of data — statistically unreliable. In the following sections we examine models that attempt to model the joint probability distribution in (3.1) by employing the notion of the *Markov property*. A stochastic process $\mathbf{x} = \{x_1, x_2, \ldots, x_T\}$ has the Markov property if

$$P(x_{t+1} | x_t, x_{t-1}, \ldots, x_1) = P(x_{t+1} | x_t) \ , \tag{3.2}$$

that is given the present state of the process, future states are conditionally independent of past states.

3.1.1 Markov Models

Markov models directly estimate the conditional probability distributions in (3.1) by employing the Markov property. In the general case, a Markov model of order n, also referred to as a *Markov chain* of order n or an *n-gram* model, assumes that a data sequence $x_{1:T}$ is generated from a stochastic process satisfying

$$P(x_t | x_{t-1}, \ldots, x_1) \equiv P(x_t | x_{t-1}, \ldots, x_{t-n}), \quad n \in \mathbb{N} \ . \tag{3.3}$$

Therefore, in a Markov model we assume that the current observation depends only on the past n observations or, put differently, given the last n observations the future is conditionally independent of the past. The number of parameters we need to estimate increases exponentially with the order of the model and typically Markov models of up to fourth order are used in real-world applications, as estimates for higher order models become unreliable.

3.1.2 Hidden Markov Models

Hidden Markov models (Rabiner and Juang, 1986) are a well-established tool for modelling sequential data. They were originally developed in the context of speech recognition, but have been successfully applied in many diverse domains, including prediction of protein structure, handwriting character recognition and financial time-series.

As seen from (3.3), n-gram models have limited flexibility in modelling temporal dependencies. By considering all n preceding observations, the model retains information that is not necessarily useful for prediction, while discarding any longer range interactions. Hidden Markov models overcome this limitation by assuming that the predictive information coming from past observations can be efficiently summarised using a discrete latent variable, often referred to as the hidden state.

In a hidden Markov model, each observation, x_t, has a corresponding hidden variable, z_t, and is conditionally independent from all previous observations given the state of the hidden variable. At the same time the sequence of hidden variables is modelled as a 1^{st} order Markov chain, so that the state of hidden variable z_t is conditionally independent from all preceding hidden variables given the state of z_{t-1}. According to these conditional independence assumptions, the joint probability distribution of an observed sequence, $\mathbf{x}$, and its corresponding hidden sequence, $\mathbf{z}$ can be factorised as

$$P(\mathbf{x}, \mathbf{z}) = P(z_1) \left[\prod_{t=2}^{T} P(z_t|z_{t-1}) \right] \prod_{t=1}^{T} P(x_t|z_t) \ . \tag{3.4}$$

An in-depth discussion of Hidden Markov Models (HMMs) is outside the scope of this work, but the interested reader can find many good tutorials and reviews in the literature. For instance, Ghahramani (2001) examines HMMs within the Bayesian Network formalism, explains algorithms for inference and learning and discusses limitations of HMMs. Bengio (1996) reviews the basics of HMMs and several extensions and generalisations including, Input-Output HMMs and switching state-space models. Finally, a more recent description of HMMs can be found in Chapter 13 of Bishop (2006), together with a description of Linear Dynamical Systems, their continuous state counterparts.

3.1.3 Variable-Length Markov Models

The Variable-Length Markov Model (VMM) (Ron et al., 1996) is a statistical model for discrete sequential data defined over a finite alphabet Σ. Similar to an n-gram model, the VMM models a sequence of observations by making specific assumptions about the conditional probability distributions in (3.1). Its advantage to a standard Markov model (*n-gram*) is that the order of the former is not fixed, but instead depends on the observed context. Long contexts that occur frequently in the data are used during prediction, while for infrequent ones, their shorter counterparts are used. This way the model can learn large and small order Markov dependencies, thus capturing contextual information of considerable complexity.

Due to the absence of any hidden variables, a variable-length Markov model of finite length can be less powerful than an HMM. However, this simplicity can be advantageous, as it removes computational and statistical complexities associated with inference of the hidden states. Therefore, VMMs can be easier and faster to learn compared to HMMs and can be more reliable when we have a limited amount of data. Additionally, they are easier to analyse and build upon, as we will see later on.

The use of variable-length Markov models for music modelling was first proposed by Dubnov et al. (2003) and was further examined in Begleiter et al. (2004). In this context, the alphabet is the set of possible notes for the melody line and each piece of music is represented as a sequence of notes. Despite its simple nature, the VMM is considered to generate state-of-the-art musical results when modelling melodies (Paiement, 2008). This is mainly attributed to the fact that the model parses music into a lexicon of motifs, thus capturing the building blocks of melodic structure.

Musical Motif

In Western Music, the smallest building block of a piece is called a motif. Motifs typically comprise three, four or more notes and most pieces can be expressed as a combination of different motifs and their transformations. Frequent transformations include replacement, splitting and merging of notes, and typically respect the metrical boundaries of a piece. Successful capturing of musical motifs can be very useful when modelling melodies, as specific motifs and their transformations are highly likely to be repeated within a piece, as well as among pieces from the

same musical form.

3.1.3.1 Learning and Prediction

Many algorithms can be used for learning a variable-length Markov model. The model is effectively defined by a set of conditional probability distributions of the form (3.3), where n on the right-hand side can vary for different contexts. Therefore, a learning algorithm specifies two elements of the resulting model. The first one is the set of contexts which are used for prediction. The selection method attempts to identify contexts that occur frequently enough in the data and convey useful information for predicting the next symbol. The length of the contexts to be considered can be bounded by a constant maximum length, or be driven by other criteria such as the frequency of occurrence.

The second element of the learning algorithm is the estimation method for the conditional probability distributions given each of the selected contexts and in particular the *smoothing* technique used. The probability estimates are always based on the counts of events occurring after each context, however, different algorithms employ different smoothing techniques to handle the "zero-frequency problem", that is the existence of events that are probable yet are not observed in the data.

In this work, we use the Probabilistic Suffix Tree (PST), which is the algorithm proposed by Ron et al. (1996) for learning a Variable-Length Markov Model. We refer the reader to Begleiter et al. (2004) for a comparison of different VMM learning algorithms, including the PST and popular algorithms from lossless compression, such as the Context Tree Weighting (CTW), Prediction by Partial Match (PPM) and the Lempel-Ziv 78 (LZ78).

The Probabilistic Suffix Tree

The PST algorithm constructs a single rooted tree encoding the VMM contexts as paths from the leaf nodes to the root. Each node in the tree is labelled by a symbol from the alphabet and is identified by the conditional probability distribution of the next symbol given the context we acquire by concatenating all labels in the path from the node to the root; the deeper a node the longer the context we acquire. The depth of the tree is upper bounded by L which denotes the

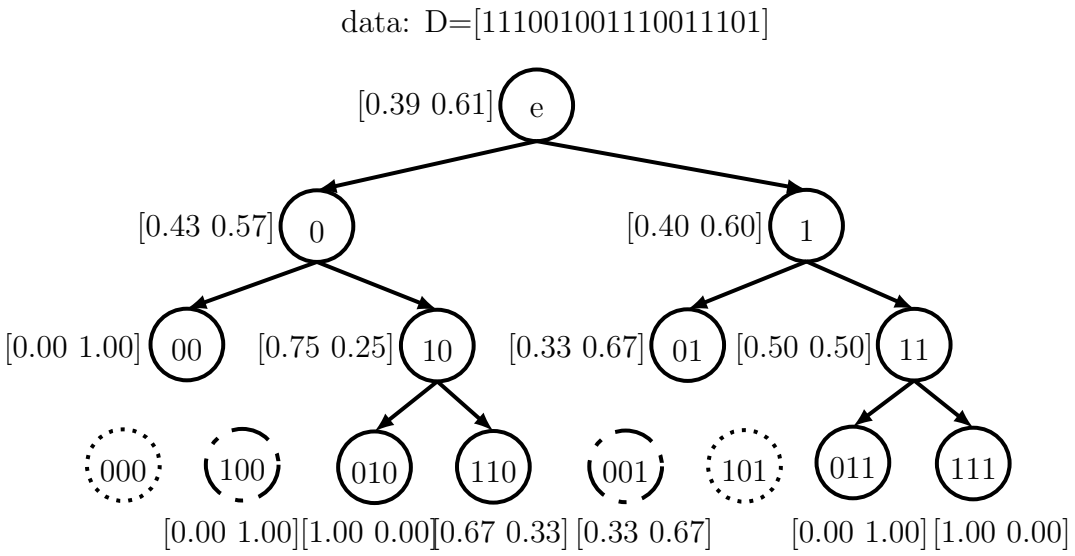


Figure 3.1: An example Probabilistic Suffix Tree for a binary sequence. For readability, nodes are labelled with the complete concatenated context. Each node is identified with the conditional probability distribution of the next symbol given the context, estimated from the empirical counts. Solid lines represent nodes included in the tree. Dotted lines represent nodes excluded during Phase 1 (their contexts are not observed frequently enough in the data). Dashed lines correspond to nodes removed during Phase 2 (their contexts do not provide additional information).

maximum allowed length for a context.

Let Σ be a finite alphabet and σ denote a symbol from the alphabet. Let j index the nodes in the tree and $\mathbf{c}_j = \sigma_\ell \sigma_{\ell-1} \dots \sigma_1$, $\ell \in \{1, 2, \dots, L\}$, denote the context corresponding to node j, that is the context we acquire by concatenating all symbols in the path from node j to the root. Finally, let $\text{suffix}(\mathbf{c}_j) = \sigma_{\ell-1} \dots \sigma_1$ be the largest suffix of context $\mathbf{c}_j$. This corresponds to the context we acquire for the parent of node j, having removed the symbol of node j from the concatenation. Learning proceeds in three phases:

- **Phase 1 – Expand the tree $\mathcal{T}$:** Starting from a single root node labelled by the empty string, perform an iterative breadth-first search and add to the tree nodes whose context satisfies the following criteria
  - The frequency counts of a context $\mathbf{c}_j$ exceed a fixed threshold ϵ_1.
  - The length of a context $\mathbf{c}_j$ is less than or equal to the maximum length L.

- **Phase 2 – Amend the tree $\mathcal{T}$:** Perform an up-pass from the leaf nodes to the root and remove nodes that do not contribute additional information relative to their parent
  - Remove node j if $\frac{1}{\epsilon_2} < \frac{P(\sigma|\mathbf{c}_j)}{P(\sigma|\text{suffix}(\mathbf{c}_j))} < \epsilon_2, \forall \sigma \in \Sigma$, with $\epsilon_2 \geq 1$.
  - If a node j satisfies the additional information criterion then all the nodes in the path from node j to the root are included in the final tree.
- **Phase 3 – Smoothing** Smooth the conditional probability distributions by adding a minimum probability γ and renormalising.

The resulting tree comprises contexts that occur frequently enough in the data and carry additional information for predicting the next symbol. The first condition makes the estimates for the conditional probability distributions reliable, while the second one reduces redundancy and makes prediction faster, by using a pruned tree. An example tree for a binary sequence is depicted in Figure 3.1. Note that for the purpose of readability, the nodes are labelled by the full contexts, having concatenated the symbol for the node with the suffix coming from its parent. We refer the reader to Ron et al. (1996) for a more detailed description and analysis of the PST algorithm.

Making Predictions

To make a prediction given an observed subsequence, we start at the root node and follow the path labelled by the observed symbols – starting from the most recent observation and going backwards in time – until we reach a leaf node, where we read the conditional probability distribution for the next symbol. Note that all the nodes are identified with a conditional probability distribution, but only the leaf nodes are used during prediction. Since the tree is not complete, this look-up procedure gives rise to both shorter and longer contexts for prediction.

3.2 The Dirichlet Variable-Length Markov Model

The Dirichlet Variable-Length Markov Model (Dirichlet-VMM) (Spiliopoulou and Storkey, 2011) is a formulation of the VMM that uses Bayesian agents to leverage all available information, while remaining analytically tractable during inference and learning. Similar to a VMM, it models the conditional probability

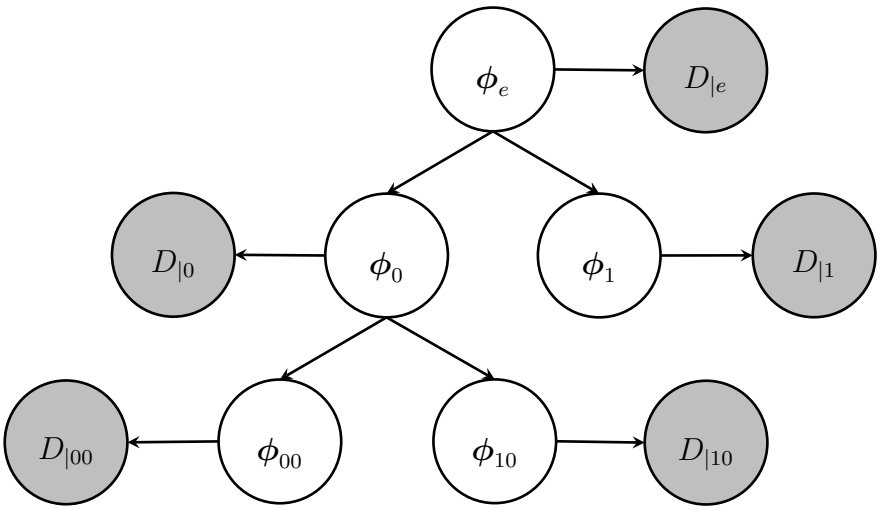


Figure 3.2: An example Dirichlet-VMM tree for binary data. The posterior distribution for the parameter vector $\boldsymbol{\phi}_j$ identifying node with context j is a Dirichlet distribution with a contribution coming from the parent node (with context suffix(j)) and a contribution coming from the data satisfying context j, denoted in the figure by $D_{|j}$.

distribution of the next symbol given a context, where the length of the context varies according to what we actually observe. However, parameter estimation is now defined through an iterative procedure that is locally Bayesian. The conditional distribution identifying each node (agent) is modelled through a prior Dirichlet distribution centred at the discrete probability vector of the parent node. Note that in the Dirichlet-VMM the data is accessed at every node. This is in contrast to a hierarchical Bayesian model, where the data would only be accessed at the leaf nodes.

The model consists of Bayesian agents that recursively consider more and more specialised information about prediction, while basing their prior belief on preceding, less refined agents. As before, let σ denote a symbol from a finite alphabet Σ and consider a suffix tree, $\mathcal{T}$, defined over Σ. Let j index the nodes in the tree, with $\mathbf{c}_j = \sigma_\ell \dots \sigma_1$ denoting the context of node j and ℓ being the depth of node j. As in the VMM, each node is identified by the conditional probability distribution of the next symbol given an already observed context. Let $\boldsymbol{\phi}_j$ denote the discrete probability vector identifying node j, with $\phi_{i|j} \equiv P(x_t = i | x_{t-\ell}, \dots, x_{t-1} = \mathbf{c}_j, \mathcal{T}), i \in \{1, \dots, \|\Sigma\|\}$ and $\sum_i^{\|\Sigma\|} \phi_{i|j} = 1$. In the Dirichlet-VMM, the discrete probability vector $\boldsymbol{\phi}_j$ is the belief of agent j about prediction. The model is defined recursively from top to bottom, so that agents

at depth ℓ base their prior beliefs on the most relevant agent from depth $\ell - 1$. This is specified through a prior Dirichlet distribution centred at the posterior belief of the parent node of each agent j, so that

$$\boldsymbol{\phi}_j | \hat{\boldsymbol{\phi}}_{\mathbf{pa}(j)} \sim \text{Dirichlet}(\beta \hat{\boldsymbol{\phi}}_{\mathbf{pa}(j)}) \ , \tag{3.5}$$

where β denotes the concentration parameter of the Dirichlet distribution, $\mathbf{pa}(j)$ denotes the parent of node j, with corresponding context $\mathbf{c}_{\mathbf{pa}(j)} = \sigma_{\ell-1} \dots \sigma_1$ and $\hat{\boldsymbol{\phi}}$ is a discrete probability vector denoting the posterior predictive distribution of an agent.

The prior Dirichlet distribution of the agent at the root node is defined using a base measure, $\boldsymbol{\phi}_b$, so that

$$\boldsymbol{\phi}_e | \boldsymbol{\phi}_b \sim \text{Dirichlet}(\beta \boldsymbol{\phi}_b) \ , \tag{3.6}$$

where e denotes the empty string.

Note that the Dirichlet-VMM is different to a hierarchical Bayesian model, as the $\hat{\boldsymbol{\phi}}_{\mathbf{pa}(j)}$'s are not specified a priori, but are instead inferred during a recursive procedure. This means that we need to look at the data and perform inference about the predictive probability of the parent node before we can specify the prior distribution for the child node. As we will see below, this recursive construction allows us to refine our beliefs about prediction in a Bayesian manner, while keeping the model analytically tractable.

3.2.1 Inference and Learning

Learning in the Dirichlet-VMM is performed recursively from top to bottom. An agent j at depth ℓ uses the belief of agent $\mathbf{pa}(j)$ at depth $\ell - 1$ as a prior and looks at the data satisfying the conditional distribution $P(x_t | x_{t-\ell}, \dots, x_{t-1} = \mathbf{c}_j, \mathcal{T})$ to make posterior inference about prediction. Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_D\}$ be a dataset comprising D sequences, with $\mathbf{x}_d = \{x_{1,d}, \dots, x_{t,d}, \dots, x_{T_d,d}\}$, where T_d denotes the length of sequence d and $x_{t,d}$ is a discrete variable taking values in $\{1, \dots, \|\Sigma\|\}$. Let $\hat{\boldsymbol{\phi}}_j$ denote the posterior predictive probability vector identifying node j after observing data $\mathbf{X}$, with $\hat{\phi}_{i|j} \equiv P(x_t = i | x_{t-\ell}, \dots, x_{t-1} = \mathbf{c}_j, \mathbf{X}, \mathcal{T})$. This is effectively the posterior belief of agent j about prediction.

Now let us examine the behaviour of the agents from top to bottom. The top most agent (root node) can make predictions for all the data — as it only conditions on the empty string e. Due to the conjugacy of the Dirichlet distribution to the categorical, the posterior distribution over $\boldsymbol{\phi}_e$ is still a Dirichlet distribution given by

$$\boldsymbol{\phi}_e|\boldsymbol{\phi}_b, \mathbf{X} \sim \text{Dirichlet}(\beta\boldsymbol{\phi}_b + N_{\cdot|e}) \ , \tag{3.7}$$

where $N_{\cdot|e}$ is a vector counting how many times each value $i \in \{1, \ldots, \|\Sigma\|\}$ has occurred in the data.

The posterior predictive distribution at the root node is given by

$$\begin{aligned}
\hat{\phi}_{i|e} = P(x_t = i|\mathbf{X}, \mathcal{T}) &= \int P(x_t = i, \boldsymbol{\phi}_e|\mathbf{X}, \mathcal{T}) \, \mathrm{d}\boldsymbol{\phi}_e \\
&= \int P(x_t = i|\boldsymbol{\phi}_e) P(\boldsymbol{\phi}_e|\mathbf{X}, \mathcal{T}) \, \mathrm{d}\boldsymbol{\phi}_e \\
&= \int \phi_{i|e} \, \text{Dirichlet}(\boldsymbol{\phi}_e|\beta\boldsymbol{\phi}_b + N_{\cdot|e}) \, \mathrm{d}\boldsymbol{\phi}_e \\
&= \mathrm{E}_{P(\boldsymbol{\phi}_e|\beta\boldsymbol{\phi}_b, \mathbf{X})} \left[\phi_{i|e}\right] \\
&= \frac{\beta\phi_{i|b} + N_{i|e}}{\sum_k \left(\beta\phi_{k|b}\right) + N} \ ,
\end{aligned} \tag{3.8}$$

where N denotes the total counts and $\mathrm{E}_{P(\cdot)}[\cdot]$ denotes an expectation taken over the distribution $P(\cdot)$.

Now consider an agent j at depth ℓ. The agent performs more refined predictions by looking at the details of the data and in particular by only considering data satisfying the conditional probability distribution $P(x_t|x_{t-\ell}, \ldots, x_{t-1} = \mathbf{c}_j)$. Due to the hierarchical construction of the contexts, the most relevant prior information available to agent j is the posterior belief of the agent sitting at node $\mathbf{pa}(j)$. This is the discrete probability vector $\hat{\boldsymbol{\phi}}_{\mathbf{pa}(j)}$ and is used as a base measure for the prior Dirichlet distribution of node j. Similar to before the posterior distribution of $\boldsymbol{\phi}_j$ is a Dirichlet distribution given by

$$\boldsymbol{\phi}_j|\hat{\boldsymbol{\phi}}_{\mathbf{pa}(j)}, \mathbf{X} \sim \text{Dirichlet}(\beta\hat{\boldsymbol{\phi}}_{\mathbf{pa}(j)} + N_{\cdot|j}) \ , \tag{3.9}$$

where $N_{\cdot|j}$ denotes the counts associated with context $\mathbf{c}_j$ in the data.

Similar to (3.8), the predictive distribution of agent j is given by an expectation

over the posterior distribution in (3.9)

$$\begin{aligned}
\hat{\phi}_{i|j} &= P(x_t = i | x_{t-\ell}, \ldots, x_{t-1} = \mathbf{c}_j, \mathbf{X}, \mathcal{T}) \\
&= \int P(x_t = i, \boldsymbol{\phi}_j | x_{t-\ell}, \ldots, x_{t-1} = \mathbf{c}_j, \mathbf{X}, \mathcal{T}) \, \mathrm{d}\boldsymbol{\phi}_j \\
&= \int P(x_t = i | \boldsymbol{\phi}_j) P(\boldsymbol{\phi}_j | \mathbf{X}, \mathcal{T}) \, \mathrm{d}\boldsymbol{\phi}_j \\
&= \int \phi_{i|j} \, \mathrm{Dirichlet}(\boldsymbol{\phi}_j | \beta \hat{\boldsymbol{\phi}}_{\mathbf{pa}(j)} + N_{\cdot|j}) \, \mathrm{d}\boldsymbol{\phi}_j \\
&= \mathrm{E}_{P(\boldsymbol{\phi}_j | \beta \boldsymbol{\phi}_{\mathbf{pa}(j)}, \mathbf{X})} \left[\phi_{i|j} \right] \\
&= \frac{\beta \hat{\phi}_{i|\mathbf{pa}(j)} + N_{i|j}}{\sum_k \left(\beta \hat{\phi}_{k|\mathbf{pa}(j)} + N_{k|j} \right)} \, .
\end{aligned} \tag{3.10}$$

Therefore, parameter learning in the Dirichlet-VMM can be performed in a single top-down pass of the tree, where we recursively compute the expectations defined by (3.10). The mean of the posterior Dirichlet distribution at each node, is the measure for the prior Dirichlet distribution at the child nodes. Note that the top levels of the hierarchy have a large amount of associated data, but as we progress down the tree the amount of data reduces. In the limit where there is no data, the posterior distribution for that node is given by the posterior distribution of the parent node.

This means that, in principle, we could consider a tree containing all the subsequences that occur in the data or, in fact, all possible subsequences derived from the alphabet Σ. Practically though, we need to use some pruning technique that results in a computationally manageable tree, without discarding much information about the data. In this work, we use the Probabilistic Suffix Tree algorithm and evaluate how the performance of the Dirichlet-VMM changes for different thresholds of the pruning criteria.

The hierarchical construction of the Dirichlet-VMM allows us to maintain information coming from the shorter contexts into the predictive probabilities of the longer ones. The parameters for the children of a node are coupled through the shared Dirichlet prior with mean probability the discrete probability vector of the parent and spread defined by the hyper-parameter β. Hence, information regarding prediction is propagated in the tree, through the recursive definition of the prior from the root to the leaf nodes.

Finally, we note that in the Dirichlet-VMM formulation we do not need to explicitly apply a smoothing technique. In contrast to a VMM, where we need to handle the "zero-frequency" problem, in the Dirichlet-VMM this is resolved in an implicit manner, through the prior distribution at the root node. In this work, we set the measure for the prior Dirichlet distribution at the root node, to a uniform measure, such that $\boldsymbol{\phi}_b = \left[\phi_{1|b} \dots \phi_{\|\Sigma\||b}\right]^{\mathsf{T}}$, with $\phi_{1|b} = \dots = \phi_{\|\Sigma\||b} = \frac{1}{\|\Sigma\|}$ and $\sum_{i=1}^{\|\Sigma\|} \phi_{i|b} = 1$. The posterior distributions become specific by looking at the data, but remain "smoothed" through the propagated contribution of the root node.

3.2.2 Relation to other models

Similar to n-gram models, the performance in the variable-length Markov models is significantly influenced by the way the predictive probabilities are smoothed. We refer the reader to Chen and Goodman (1998) for a systematic evaluation of smoothing algorithms for n-gram models. They present an extensive empirical comparison of several smoothing techniques – including Jelinek-Mercer smoothing and the more advanced Kneser-Ney and modified Kneser-Ney smoothing – assessing their performance on different language datasets.

From Equation (3.10) we can see that the inference procedure in the Dirichlet-VMM is a form of Jelinek-Mercer smoothing: the predictive distribution for context j is a linear interpolation between the Maximum Likelihood model for context j and the predictive distribution for context $\mathbf{pa}(j)$. Starting from Equation (3.10) we have

$$\hat{\phi}_{i|j} = \frac{\beta\hat{\phi}_{i|\mathbf{pa}(j)} + N_{i|j}}{\sum_k \left(\beta\hat{\phi}_{k|\mathbf{pa}(j)} + N_{k|j}\right)} = \frac{\beta}{\beta + N_j}\hat{\phi}_{i|\mathbf{pa}(j)} + \frac{N_{i|j}}{\beta + N_j} \ , \tag{3.11}$$

where $N_j = \sum_k N_{k|j}$ is the number of times context j appears in the data. If we define

$$\lambda_j = \frac{\beta}{\beta + N_j} \ , \tag{3.12}$$

then we can write

$$\hat{\phi}_{i|j} = \lambda_j\hat{\phi}_{i|\mathbf{pa}(j)} + (1 - \lambda_j)\phi^{\mathrm{ML}}_{i|j} \ , \tag{3.13}$$

where $\phi^{\mathrm{ML}}_{i|j} = \frac{N_{i|j}}{N_j}$ is the Maximum Likelihood estimate for the predictive probability of symbol i given context j.

In Jelinek-Mercer smoothing the λ_j's are set using a cross-validation procedure called 'deleted interpolation'. In principle, we can have a different λ_j for each context j. However, this is impractical, as estimating all the λ_j's independently would require an excessive amount of data. In deleted interpolation, the λ_j's are partitioned into a moderate number of buckets, and the λ_j's in each bucket are constrained to the same value. The partitioning is performed with respect to N_j, so that each bucket contains contexts with similar number of counts. The interpolation weights we acquire in the Dirichlet-VMM also depend on the counts of each context, which is noticeable by our definition of λ_j in (3.12).

The Dirichlet-VMM is related to the hierarchical Dirichlet language model of Mackay and Peto (1995), who also use a Dirichlet distribution prior for the predictive distributions of an n-gram model. Here, instead of fixed-order statistics we use variable-order ones. Additionally, we do not infer the measure of the Dirichlet prior, but instead successively tie it to the smoothed predictive probability for the shorter context. Therefore, in contrast to the hierarchical Bayesian model of Mackay and Peto (1995), where the data is only accessible at the last stage of the hierarchy, in the Dirichlet-VMM every agent (node) j is associated with data satisfying the conditional distribution $P(x_t = i|x_{t-\ell}, \ldots, x_{t-1} = \mathbf{c}_j)$.

The Dirichlet-VMM is also related to the sequence memoizer (Wood et al., 2009), which is a hierarchical Bayesian nonparametric model that uses an unbounded-depth Pitman-Yor process to specify the prior over the predictive distributions. The model has an infinite number of predictive distributions and uses the coagulation and fragmentation operators to marginalise out nodes (variables) that are not of interest and thus get a collapsed graphical model. In the collapsed model, the predictive distributions can be approximated using samples from a Chinese Restaurant Process (CRP), which can be obtained through a Markov chain Monte Carlo sampling scheme as described in Teh et al. (2006). On the contrary, the finite nature of the Dirichlet-VMM and its iterative prior specification that uses local Bayesian agents, makes the inference procedure entirely conjugate and thus allows us to analytically compute the posterior predictive distributions. Interestingly, Teh (2006) shows that an n-gram model with a hierarchical Pitman-Yor process prior is related to the interpolated Kneser-Ney smoothing technique, where an approximation to the former recovers an exact formulation of the latter.

3.3 Combining Global and Local Models

In this section we examine how to extend the Dirichlet-VMM in order to efficiently deal with novelty. This is an important aspect when modelling music, as composition is a creative process, and though structure is evident in almost every musical piece, so is originality and manifestation of new music ideas and thus previously unseen combinations of music elements. As a result the intra (within) and inter-piece (among) structure of music has different properties.

For instance, specific novel elements are likely to be repeated within a piece, in a composer's effort to elaborate the musical idea or running theme of the piece. Therefore, we might expect longer constructs or specific transitions to be repeated more frequently within a piece than across pieces, whereas frequent motifs and shorter phrases are likely to be shared throughout a genre, with different arrangements. Identifying such novel musical constructs and efficiently incorporating this information in a model can lead to improved performance in music tasks and help distinguish the inter and intra piece structural differences.

A typical approach used in the Natural Language Processing (NLP) literature, is to learn an n-gram model using the training corpus and to augment the model dynamically during testing, with information coming from the already processed test data. Here, we address the aspect of novelty in a similar manner. We consider two sources of available information; the training corpus, which is available in its entirety and the already processed part of a testing piece, which becomes available in an online fashion.

In the following three Sections we examine different ways of combining these two data sources. We often refer to models as global or local, depending on which source of information is used during learning. A global model is learned once using the entire training corpus and therefore captures statistical dependencies in the whole music genre. A local model is learned dynamically as we process a testing piece, thus representing the specific structure of a single music sequence.

3.3.1 Update the counts

The first combination we examine incorporates information coming from a test piece by dynamically updating the counts in the model as we observe events in the

Algorithm 1 Parameter estimation in the GL_1 coupling scheme.

Input: $\mathcal{T}^G$, $\mathbf{X}^G$, $\mathbf{x}^L$

Initialise: $\mathcal{T}^{GL_1} = \mathcal{T}^G$

Set: useDynamicStructure to True or False according to whether a dynamic tree structure is to be used (See text in Section 3.3.1 for details).

for each time-step t in test sequence $\mathbf{x}^L$ **do**

Compute Predictive Distribution:

$$P(x_t = i|x_{t-\ell}, \ldots, x_{t-1} = \mathbf{c}_j, \mathbf{X}^G, \mathbf{x}^L_{<t}, \mathcal{T}^{GL_1}) = \frac{\beta \mathrm{E}[\phi^{GL_1}_{i|\mathbf{pa}(j)}] + N^G_{i|j} + \{N^L_{i|j}\}_{<t}}{\beta + \sum_k \left(N^G_{k|j} + \{N^L_{k|j}\}_{<t}\right)}$$

Dynamically update the tree:

while $\mathbf{c}_j$ has suffix **do**

if $j \notin \mathcal{T}^{GL_1}$ AND useDynamicStructure **then**

Add j in $\mathcal{T}^{GL_1}$

Set $N^G_{\cdot|j} = \mathbf{0}$

end if

Set $\{N^L_{x_t|j}\}_{<t+1} = \{N^L_{x_t|j}\}_{<t} + 1$

Set $\mathbf{c}_j = \mathrm{suffix}(\mathbf{c}_j)$

end while

end for

test sequence. Let superscripts G and L denote global (training data) and local (already observed testing sequence) information, respectively, and GL denote the combined information. For each testing sequence we define the GL_1-Dirichlet-VMM as

$$\boldsymbol{\phi}^{GL_1}_j \sim \mathrm{Dirichlet}(\beta \hat{\boldsymbol{\phi}}^{GL_1}_{\mathbf{pa}(j)}) \ . \tag{3.14}$$

Therefore, the prior formulation in this scheme is the same as in the standard Dirichlet-VMM. However, inference of the posterior distributions for the combined model is performed dynamically, as we observe a testing sequence. Let $\mathbf{X}^G$ be the set of training sequences, and $\mathbf{x}^L$ be a new sequence that we are trying to predict. Also let $\mathbf{x}^L_{<t}$ be the part of the testing sequence we have already processed, with corresponding counts denoted by $\{N^L_{\cdot|\cdot}\}_{<t}$. Each node is now associated with data coming from both the training set and the already processed test sequence, such that

$$\boldsymbol{\phi}^{GL_1}_j | \mathbf{X}, \mathbf{x}^L_{<t}, \boldsymbol{\phi}^{GL_1}_{\mathbf{pa}(j)} \sim \mathrm{Dirichlet}\left(\beta \mathrm{E}[\boldsymbol{\phi}^{GL_1}_{\mathbf{pa}(j)}] + N^G_{\cdot|j} + \{N^L_{\cdot|j}\}_{<t}\right) \ . \tag{3.15}$$

Therefore, parameter inference under this coupling scheme, places equal importance to observations coming from the testing sequence and the training data. This formulation has two limitations. Firstly, in the limit of infinite training data, the observations from the testing sequence will not have any impact on parameter inference. Perhaps more importantly, the result of the inference procedure is more sensitive to the choice of the training data than usual, because the choice of the training data additionally determines the "weighting" of the testing data. More training data will make observations from the testing sequence less important, and a different choice of training sequences can result in different observations from the testing sequence becoming more or less important during inference.

With respect to the tree structure of the model we examine two possibilities. In the first case, which we refer to as "fixed tree structure", we build the tree once using the PST algorithm on the training data and its structure remains unchanged during testing. In the second case, which we refer to as "dynamic tree structure", for every new testing sequence we start with the aforementioned (global) tree structure and allow new contexts from the already processed sequence to be added dynamically. The global counts of new contexts are set to the zero vector, i.e. for a new context j', we have $N^G_{\cdot|j'} = \mathbf{0}$. However, the conditional probability distributions identifying new contexts are still influenced by the global model through the parental prior.

Algorithm 1 sketches the procedure for parameter estimation in the GL_1-Dirichlet-VMM. Note that the if-statement of the algorithm is only executed in the case of the 'dynamic tree structure' described above.

3.3.2 Product Formulation

In this formulation the global and local models are learned independently and are combined during prediction through a product. More specifically, let G denote the global model, learned using the training data, and L denote a local model, learned dynamically while predicting a test sequence, then we have

$$P(x_t = i | x_{t-\ell}, \ldots, x_{t-1} = \mathbf{c}_j, \hat{\boldsymbol{\phi}}^G_j, \hat{\boldsymbol{\phi}}^L_j) = \frac{\hat{\phi}^L_{i|j}\hat{\phi}^G_{i|j}}{\sum_k \hat{\phi}^L_{k|j}\hat{\phi}^G_{k|j}} \ , \tag{3.16}$$

where $\hat{\boldsymbol{\phi}}_j^G$ and $\hat{\boldsymbol{\phi}}_j^L$ are the posterior estimates for the predictive distributions of the global and the local Dirichlet-VMM, respectively, defined according to (3.10). In this case we have two experts with information regarding a music piece. The information encoded by the two experts is combined through a product, which means that a configuration must be consistent with both the global and the local model, in order to have high probability.

Parameter estimation of the global model is performed once using all the training sequences $\mathbf{X}^G$. On the other hand, the parameters for the local model and the predictive distribution in (3.16) are updated dynamically as we observe the test sequence, each time using the already processed part of the sequence $\mathbf{x}_{<t}^L$. We refer to this coupling scheme as GL_2.

3.3.3 Mixing Priors

In this case, we combine the global and local models through a "mixing" formulation. The predictive distribution takes into account the counts coming from the test sequence and two prior contributions; a parental measure defined through a local Dirichlet-VMM, and a node measure defined through a global Dirichlet-VMM. More specifically, we have

$$P(x_t=i|x_{t-\ell},\ldots,x_{t-1}=\mathbf{c}_j,\mathbf{x}_{<t}^L,\mathcal{T}^G,\mathcal{T}^L)=\frac{\beta_0\hat{\phi}_{i|j}^G+\beta_1\hat{\phi}_{i|\mathbf{pa}(j)}^L+\{N_{i|j}^L\}_{<t}}{\beta_0+\beta_1+\sum_k\{N_{k|j}^L\}_{<t}} \quad . \quad (3.17)$$

Similar to the product formulation, the global model is learned once, whereas the local model and predictive distributions are updated dynamically after each time-step is processed.

Note that we have not defined a prior distribution that would lead to the posterior predictive distribution in (3.17). However, we think that the coupling scheme in this predictive distribution is interesting, as it is based on additive contributions and treats information coming from the test sequence and the training data disjointly. Therefore, using this predictive distribution we can examine how performance changes as we give more or less importance to the different factors,

that is the contribution from the global model, the parental prior contribution from the local model and the counts coming from the testing sequence.

3.4 Model Evaluation

In this section we analyse the performance of the VMM, the Dirichlet-VMM and the Global-Local model combinations of the Dirichlet-VMM. In all the experiments the tree structures are constructed using Algorithm 2. Parameter estimation is then performed using additive smoothing in the case of the VMM (Section 3.1.3.1) and inference using the Bayesian agents in the case of the Dirichlet-VMM and the Global-Local combinations (Sections 3.2.1 and 3.3.1 – 3.3.3). In this section we report results from performing 10-fold cross-validation on 200 pieces. In the next section we present a comparative analysis, where we select the best performing setting for each of the models and evaluate performance on a held-out testing set.

3.4.1 Analysing Meta and Hyper-Parameters

First we examine the performance of the models with respect to different settings of the meta-parameters for the tree construction algorithm and the hyper-parameter β of the prior Dirichlet distribution.

3.4.1.1 Variable-Length Markov model

For the Variable-Length Markov model, the meta-parameters we examine are the maximum length of a context, L, the frequency counts threshold, ϵ_1, the additional information threshold, ϵ_2, and the minimum probability, γ, used for smoothing. To set these parameters we perform the 10-fold cross-validation procedure (see section 2.3.1 for details) over the product space $L \in (1, 5, 10, 20, 50) \times \epsilon_1 \in (10^{-3}, 5 * 10^{-4}, 10^{-4}) \times \epsilon_2 \in (1, 2, 5, 10, 20) \times \gamma \in (10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-16})$. Table 3.1 shows the average next-step prediction log-likelihood under different VMMs. We only present a subset of the results and note that the full product space has many settings of the meta-parameters that are redundant. For instance, further increasing the maximum allowed depth to $L = 50$ results in the same tree structures as using $L = 20$ and is thus omitted.

Algorithm 2 Build a Probabilistic Suffix Tree

Input: $L, \epsilon_1, \epsilon_2$, $\mathbf{X}$

Initialise $\mathcal{T}$ with a single root node labelled by the empty string

Set $\ell = 1$

while $\ell < L$ **do**

`Frequency Counts:` Add to $\mathcal{T}$ all nodes j that satisfy $\frac{\sum_i N_{i|j}}{\sum_{j'} \sum_i N_{i|j'}} > \epsilon_1$, where j' indexes nodes with context length same as $\mathbf{c}_j$.

Set $f_{i|j} = \frac{N_{i|j}}{\sum_i N_{i|j}}$, $i \in \{1, \ldots, \|\Sigma\|\}$.

Set $\ell = \ell + 1$.

end while

Set $\ell = L$.

while $\ell > 1$ **do**

`Additional Information:` Keep in $\mathcal{T}$ only nodes j that satisfy There exists $i \in \{1, \ldots, \|\Sigma\|\}$, such that $\frac{f_{i|j}}{f_{i|\mathbf{pa}(j)}} > \epsilon_2$ or $\frac{f_{i|j}}{f_{i|\mathbf{pa}(j)}} < \frac{1}{\epsilon_2}$.

if Node j is maintained **then**

Maintain all nodes in the path from the root to node j.

end if

Set $\ell = \ell - 1$.

end while

Looking at Table 3.1 we can further see that the effective dimensionality of the meta-parameter space is low. The important meta-parameters for maximising the log-likelihood in the reels dataset are the maximum allowed depth, L, and the frequency counts criterion, ϵ_1. On the other hand, the threshold ϵ_2 – which prunes nodes that do not contribute additional information to the tree – and the smoothing parameter γ have very little or no effect on performance. This suggests that grid-search is inefficient at covering the sub-space important to this optimisation and alternative techniques, such as random search (Bergstra and Bengio, 2012) or global optimisation should be considered (see Section 2.3.2 for a more detailed discussion).

The best performance is achieved by the VMM that has the "tightest" thresholds for both criteria controlling the size of the tree structure ($\epsilon_1 = 10^{-3}$ and $\epsilon_2 = 20$), a maximum depth of $L = 10$ and the highest value for the smoothing parameter, $\gamma = 10^{-2}$. The larger ϵ_1 means that a context must occur more times in the data before it is included in the tree. For the reels dataset, the 10^{-3} threshold

Table 3.1: Average next-step prediction log-likelihood under the VMM. Results from 10-fold cross-validation. Boxed: Best overall configuration. Bold: Best configuration for each level of the additive smoothing γ. Performance increases substantially when we move from first-order to higher order dependencies. Pruning the tree according to the frequency counts criterion (ϵ_1) has a considerable effect on performance, with more pruning leading to better results. See text for more details.

		$\epsilon_1 = 10^{-3}$			$\epsilon_1 = 10^{-4}$		
		$\epsilon_2 = 1$	$\epsilon_2 = 10$	$\epsilon_2 = 20$	$\epsilon_2 = 1$	$\epsilon_2 = 10$	$\epsilon_2 = 20$
$\gamma = 10^{-2}$	$L = 1$	-2.0887	-2.0908	-2.0908	-2.0859	-2.0880	-2.0880
	$L = 5$	-1.7958	-1.7952	-1.7953	-1.8524	-1.8522	-1.8527
	$L = 10$	-1.7933	-1.7927	**$\boxed{\textbf{-1.7925}}$**	-1.9930	-1.9899	-1.9899
	$L = 20$	-1.7933	-1.7927	-1.7925	-1.9941	-1.9912	-1.9912
$\gamma = 10^{-3}$	$L = 1$	-2.1028	-2.1046	-2.1046	-2.1000	-2.1018	-2.1018
	$L = 5$	-1.8094	-1.8089	-1.8090	-1.8601	-1.8600	-1.8605
	$L = 10$	-1.8068	-1.8063	**-1.8061**	-1.9927	-1.9896	-1.9896
	$L = 20$	-1.8068	-1.8063	-1.8061	-1.9937	-1.9908	-1.9908
$\gamma = 10^{-4}$	$L = 1$	-2.1042	-2.1060	-2.1060	-2.1014	-2.1032	-2.1032
	$L = 5$	-1.8108	-1.8103	-1.8103	-1.8610	-1.8609	-1.8613
	$L = 10$	-1.8082	-1.8076	**-1.8075**	-1.9928	-1.9896	-1.9896
	$L = 20$	-1.8081	-1.8076	-1.8075	-1.9938	-1.9909	-1.9909
$\gamma = 10^{-5}$	$L = 1$	-2.1043	-2.1061	-2.1061	-2.1015	-2.1033	-2.1033
	$L = 5$	-1.8109	-1.8104	-1.8105	-1.8610	-1.8609	-1.8614
	$L = 10$	-1.8083	-1.8078	**-1.8076**	-1.9928	-1.9896	-1.9896
	$L = 20$	-1.8083	-1.8078	-1.8076	-1.9938	-1.9909	-1.9909

means that a context must be observed roughly 50 times before it is included in the tree, whereas 10^{-4} means that we need to observe a context roughly 5 times. A maximum depth of $L=10$ means that the longest allowable context is 10 time-steps long. Therefore, the optimal VMM learns a dictionary of musical motifs, with the longest phrases stored in the tree spanning roughly a musical measure (bar). Remember that the melodies we consider have 4/4 meter and we discretise each piece into eighth notes. Hence a musical measure corresponds to 8 time-steps in our representation.

Looking more closely at the effect of the maximum depth, L, we can see that, in most cases, including longer contexts leads to increased performance. The improvement is always high when we move from first-order dependencies ($L = 1$) to fifth-order ones ($L = 5$), i.e. when we move from considering only the previous

time-step to considering short phrases of about half a measure. Remember that the length of the resulting tree also depends on the frequency counts criterion, ϵ_1, as longer contexts tend to occur on fewer occasions than shorter ones. For this dataset, the larger ϵ_1 value results in trees with depth less than or equal to 10, whereas the smaller value results in deeper trees if $L > 10$. In the former case ($\epsilon_1 = 10^{-3}$) performance remains roughly the same as we increase the L to more than 5 time-steps. However, when $\epsilon_1 = 10^{-4}$ increasing the maximum depth leads to decreased performance. When we allow less frequent contexts in the tree we have many predictive distributions for which the VMM estimation is unreliable, as additive smoothing is not flexible enough to overcome the problem of limited data.

Finally, as already mentioned the ϵ_2 and γ meta-parameters have little effect on performance. When we use $\epsilon_2 > 1$, we require that a longer context predicts something different than its shorter counterpart and prune nodes whose predictive distribution is very similar to that of their parent node (largest suffix). Our evaluation suggests that in this application reducing the redundancy in the predictive distributions of the tree is not important for maximising the data likelihood. Additionally, although dealing with limited information is important, changing the parameter for the additive smoothing has little effect, which suggests that a more flexible smoothing method is needed if we want to use longer contexts.

3.4.1.2 Dirichlet Variable-Length Markov model

For the Dirichlet-VMM, the meta-parameters we examine are the maximum length of a context, L, and the frequency counts threshold, ϵ_1. Additionally, we examine the hyper-parameter β corresponding to the concentration parameter of the prior Dirichlet distribution. Note that in this case, we construct the tree using the PST algorithm for computational efficiency. In principle, we could have a tree structure representing all the contexts that occur in the data – or even all possible contexts – as parameter learning is driven by Bayesian inference. For contexts with very few, or no, observations we would have a high contribution from the prior. However, the PST algorithm results in a smaller tree, while discarding very little information, mainly associated with contexts that occur rarely.

Table 3.2 shows the average next-step prediction log-likelihood under different

Table 3.2: Average next-step prediction log-likelihood under the Dirichlet-VMM. Results from 10-fold cross-validation. Boxed: Best overall configuration. Bold: Best configuration for each value of the hyperparameter β. Performance is optimal when we use all available information (large L/small ϵ_1) and account for the uncertainty through a fairly high prior contribution (large β). See text for more details.

		$\epsilon_1 = 10^{-3}$	$\epsilon_1 = 5\times10^{-4}$	$\epsilon_1 = 10^{-4}$
$\beta = 5$	$L = 1$	-1.9430	-1.9410	-1.9403
	$L = 5$	**-1.7531**	-1.7742	-1.8018
	$L = 10$	-1.7724	-1.8389	-2.0344
	$L = 20$	-1.7725	-1.8391	-2.0380
$\beta = 10$	$L = 1$	-1.9426	-1.9406	-1.9400
	$L = 5$	**-1.7214**	-1.7239	-1.7246
	$L = 10$	-1.7353	-1.7666	-1.8549
	$L = 20$	-1.7353	-1.7667	-1.8567
$\beta = 50$	$L = 1$	-1.9434	-1.9418	-1.9414
	$L = 5$	-1.6624	-1.6429	**-1.6240**
	$L = 10$	-1.6647	-1.6479	-1.6306
	$L = 20$	-1.6648	-1.6479	-1.6307
$\beta = 100$	$L = 1$	-1.9456	-1.9444	-1.9441
	$L = 5$	-1.6531	-1.6341	-1.6176
	$L = 10$	-1.6513	-1.6302	-1.6068
	$L = 20$	-1.6513	-1.6302	**-1.6066**
$\beta = 200$	$L = 1$	-1.9509	-1.9501	-1.9499
	$L = 5$	-1.6602	-1.6448	-1.6323
	$L = 10$	-1.6557	-1.6364	-1.6153
	$L = 20$	-1.6557	-1.6364	**-1.6151**

Dirichlet-VMMs. Again we present an indicative subset of the cross-validation results. The full product space in this case is $L \in (1, 5, 10, 20, 50) \times \epsilon_1 \in (10^{-3}, 5 * 10^{-4}, 10^{-4}) \times \beta \in (5, 10, 50, 100, 200)$. The threshold for the "additional information" criterion ϵ_2, is set to 1 in all the experiments, as it is found to have little effect in the case of the VMM.

The best next-step prediction log-likelihood is given for $\beta = 100$, $\epsilon_1 = 10^{-4}$ and $L = 20$. Similar to the VMM results, the biggest improvement in log-likelihood is attained when we increase the maximum depth, L, from 1 to 5, i.e. when we move from considering only the previous time-step to considering short phrases of about half a musical measure (bar). However, in contrast to the VMM results, in the

Dirichlet-VMM further increasing the maximum depth leads to either increased or the same performance, provided that we use a large enough concentration parameter for the prior Dirichlet distribution.

More specifically, for high enough values of the concentration parameter ($\beta \geq 50$), the Dirichlet-VMM gives better predictions when we use all the available information from the data, i.e. when we allow both longer ($L > 5$) and less frequent (smaller ϵ_1) contexts to be stored in the tree, with the improvement being more noticeable in the latter case. A higher concentration parameter means that our uncertainty regarding the training data can be adequately handled by the prior distribution. Therefore, in the Dirichlet-VMM performance is optimised when we consider all available information from the data – albeit noisy – and we use a strong prior contribution to overcome the uncertainty in the predictive probabilities of contexts that occur less frequently. To understand this better, remember that with $\epsilon_1 = 10^{-4}$, any context that occurs more than 5 times in the training data is included in the tree. However, for a context that occurs 5 times and with $\beta = 50$, the prior measure (predictive probability of the parent node) contributes roughly 10 times more than the context counts to the predictive probability distribution for that context.

On the other hand, for small values of the concentration parameter ($\beta \in \{5, 10\}$), the Dirichlet-VMM gives better predictions if we only consider the most frequent contexts, i.e. for $\epsilon_1 = 10^{-3}$. This is to be expected, as a small β value means that the contribution of the prior is weaker and thus cannot alleviate the uncertainty associated with less frequent contexts. Additionally, we can observe that for small β values, performance is highest for $L = 5$. This can also be attributed to the weak contribution of the prior, with the predictive distributions for longer contexts being too peaked on the training data and not able to generalise well. Finally, we can observe that increasing the value of the concentration parameter is beneficial only up to a certain level. For all the combinations of L and ϵ_1, the highest log-likelihood is achieved with $\beta = 100$ and performance decreases when we use $\beta = 200$. In the latter case the contribution from the prior is too strong and suppresses the information coming from the data.

3.4.1.3 Global-Local Dirichlet-VMM

Next we evaluate the three ways of combining global and local information in the Dirichlet-VMM, described in Section 3.3. Table 3.3 shows the average next-step prediction log-likelihood of Dirichlet-VMM models trained using the GL_1 combination (update the counts). The combination is evaluated using both a fixed and a dynamic tree structure. In the first case, the tree structure for each testing sequence (local model) is fixed to the tree structure of the global model. In the second case, the tree structure changes dynamically by adding new contexts that we observe during prediction. The experiments using a fixed tree structure can help elucidate whether during prediction we encounter new music constructs for which we have limited or no information, or if the "old" constructs captured by the global model have a slightly different meaning within a new piece; for instance, motifs that have a certain function, which is consistent throughout a piece, but somewhat different in different pieces.

By comparing Table 3.3 to Table 3.2 (global Dirichlet-VMM), we can see that including local information using the GL_1 combination always improves performance. For all the settings of the learning parameters, the prediction log-likelihood is higher under the GL_1-Dirichlet-VMM than the Dirichlet-VMM. The improvement is marginal when we only consider first-order dependencies ($L = 1$), but is considerable if we consider longer contexts. This is true for both fixed and dynamic tree structures, though in the latter case the improvement is even higher. This suggest that novelty is present in the form of both new musical phrases – which are added to the tree when we use a dynamic tree structure – and adapted usage of the old phrases – whose predictive probabilities are re-estimated.

Using a fixed tree structure the best performance is acquired for $L = 20$, $\epsilon_1 = 10^{-4}$ and $\beta = 50$. Similarly to the global Dirichlet-VMM, longer and wider trees result in better performance, provided that we have a large enough concentration parameter for the prior distribution. However, in this case the best performance is attained with a lower concentration parameter compared to the global Dirichlet-VMM (optimal $\beta = 50$ and $\beta = 100$, respectively). Predictive information coming from the testing piece is more accurate and thus a smaller contribution from the prior is needed.

Table 3.3: Average next-step prediction log-likelihood under the GL_1 combination of the Dirichlet-VMM (adding counts). Results from 10-fold cross-validation. Boxed: Best overall configuration for each tree structure. Bold: Best configuration for each value of the hyperparameter β. Longer and wider trees coupled with a moderate prior contribution are best when using a fixed tree structure. Narrower trees with a small concentration parameter are optimal for the dynamic tree structure. See text for more details.

		Fixed Tree Structure			Dynamic Tree Structure		
β	L	$\epsilon_1 = 10^{-3}$	$\epsilon_1 = 5\times10^{-4}$	$\epsilon_1 = 10^{-4}$	$\epsilon_1 = 10^{-3}$	$\epsilon_1 = 5\times10^{-4}$	$\epsilon_1 = 10^{-4}$
5	1	−1.9356	−1.9332	−1.9324	−1.9332	−1.9327	−1.9324
	5	−1.6391	−1.6079	**-1.5561**	−1.3741	−1.4336	−1.4877
	10	−1.6482	−1.6326	−1.6090	−1.0538	−1.1031	−1.2371
	20	−1.6483	−1.6328	−1.6116	**-1.0312**	−1.0660	−1.1597
10	1	−1.9356	−1.9332	−1.9324	−1.9336	−1.9328	−1.9324
	5	−1.6238	−1.5857	**-1.5287**	−1.4217	−1.4546	−1.4778
	10	−1.6294	−1.5979	−1.5327	−1.1228	−1.1532	−1.2263
	20	−1.6294	−1.5980	−1.5339	**-1.0909**	−1.1016	−1.1135
50	1	−1.9372	−1.9353	−1.9348	−1.9363	−1.9352	−1.9348
	5	−1.6018	−1.5632	−1.5206	−1.5178	−1.5099	−1.4997
	10	−1.5996	−1.5547	−1.4791	−1.3415	−1.3342	−1.3280
	20	−1.5997	−1.5547	**-1.4789**	−1.3115	−1.2863	**-1.2155**
100	1	−1.9399	−1.9384	−1.9381	−1.9393	−1.9383	−1.9381
	5	−1.6067	−1.5750	−1.5432	−1.5540	−1.5415	−1.5299
	10	−1.6018	−1.5623	−1.5022	−1.4291	−1.4152	−1.3999
	20	−1.6018	−1.5623	**-1.5019**	−1.4056	−1.3779	**-1.3096**
200	1	−1.9457	−1.9447	−1.9445	−1.9453	−1.9446	−1.9445
	5	−1.6262	−1.6027	−1.5807	−1.5947	−1.5825	−1.5725
	10	−1.6195	−1.5888	−1.5451	−1.5097	−1.4949	−1.4786
	20	−1.6195	−1.5888	**-1.5449**	−1.4929	−1.4681	**-1.4120**

For the dynamic tree structures, performance is highest for $L = 20$, $\epsilon_1 = 10^{-3}$ and $\beta = 5$. The parental prior has a very small contribution, allowing information coming from the testing piece to drive prediction. A larger ϵ_1 with the small concentration parameter ($\beta = 5$) decreases performance, as less frequent contexts are included in the tree and the contribution of the prior is small, thus introducing uncertainty in prediction using these contexts. However, note that for higher concentration parameters ($\beta > 10$), including less frequent contexts (smaller ϵ_1) increases the log-likelihood.

In Table 3.4 we present an indicative subset of experiments for the product formulation of combining global and local information. In this case the global and local models are learned independently and are subsequently combined by taking their product and renormalising. We use a different concentration parameter for the Dirichlet prior in each model, which we denote by β_G and β_L for the global and local models respectively. Note that in this case, the tree structure of the local model is built dynamically as we observe the testing piece.

Table 3.4: Average next-step prediction log-likelihood under the GL_2 combination of the Dirichlet-VMM (product). Results from 10-fold cross-validation. Boxed: Best overall configuration. Bold: Best configuration for each value of the hyperparameter β_G for the prior distributions in the global model. Performance is optimised with high values for both concentration parameters and narrower trees, as the product formulation penalises highly peaked predictive distributions in the experts. See text for more details.

		$\epsilon_1 = 10^{-3}$			$\epsilon_1 = 10^{-4}$		
		$\beta_L = 5$	$\beta_L = 50$	$\beta_L = 200$	$\beta_L = 5$	$\beta_L = 50$	$\beta_L = 200$
$\beta_G = 5$	$L = 1$	-1.9883	-1.8835	-1.8562	-1.9879	-1.8827	-1.8553
	$L = 5$	-1.7292	-1.5900	-1.5696	-1.8733	-1.7693	-1.7597
	$L = 10$	-1.6854	-1.4880	-1.4555	-2.0866	-2.0455	-2.0733
	$L = 20$	-1.6818	-1.4716	**-1.4350**	-2.0245	-1.9225	-1.9494
$\beta_G = 10$	$L = 1$	-1.9877	-1.8828	-1.8556	-1.9870	-1.8818	-1.8544
	$L = 5$	-1.6986	-1.5583	-1.5376	-1.7927	-1.6779	-1.6648
	$L = 10$	-1.6528	-1.4515	-1.4181	-1.9148	-1.8299	-1.8433
	$L = 20$	-1.6504	-1.4354	**-1.3978**	-1.8795	-1.7279	-1.7329
$\beta_G = 50$	$L = 1$	-1.9873	-1.8824	-1.8552	-1.9865	-1.8813	-1.8539
	$L = 5$	-1.6810	-1.5398	-1.5189	-1.7500	-1.6286	-1.6134
	$L = 10$	-1.6344	-1.4304	-1.3964	-1.8285	-1.7184	-1.7229
	$L = 20$	-1.6327	-1.4146	**-1.3762**	-1.8064	-1.6287	-1.6210
$\beta_G = 100$	$L = 1$	-1.9870	-1.8822	-1.8549	-1.9861	-1.8810	-1.8536
	$L = 5$	-1.6687	-1.5267	-1.5057	-1.7219	-1.5959	-1.5792
	$L = 10$	-1.6217	-1.4157	-1.3811	-1.7739	-1.6469	-1.6451
	$L = 20$	-1.6205	-1.4001	**-1.3611**	-1.7602	-1.5654	-1.5492
$\beta_G = 200$	$L = 1$	-1.9868	-1.8819	-1.8547	-1.9859	-1.8807	-1.8533
	$L = 5$	-1.6592	-1.5167	-1.4955	-1.7014	-1.5720	-1.5541
	$L = 10$	-1.6121	-1.4044	-1.3694	-1.7356	-1.5960	-1.5894
	$L = 20$	-1.6113	-1.3890	**-1.3495**	-1.7278	-1.5207	-1.4981

The first thing we can observe is that in the GL_2 combination, performance increases as we increase the value of the concentration parameter for the local model. Since we learn the local model independently of the global one, the available counts information in the former is limited and thus we need a strong contribution of the parental prior in order to overcome the uncertainty regarding predictions. As already mentioned, for a configuration to have high probability under the product formulation, it needs to be likely under both experts. In the local model, due to the limited information the predictive distributions can be too peaked, leading to decreased performance, unless the contribution of the prior is strong enough to overcome this problem.

Higher concentration parameters for the global model and pruning the tree using the frequency counts criterion, i.e. excluding contexts for which we have limited information, also lead to increased performance. Again, these results can be justified by the fact that the product formulation tends to prefer flatter distributions; if one expert "likes" a configuration a lot, but the other one does not, the model is penalised. Therefore, contexts with limited data and weak prior contributions tend to worsen performance.

Finally, we can note that with an appropriately high concentration parameter for the local model, the GL_2 combination has better performance than using the equivalent global Dirichlet-VMM.

Tables 3.5 and 3.6 show the next step prediction log-likelihood under the GL_3 combination (mixing the priors) for an indicative subset of experiments using a fixed and a dynamic tree structure for the local model, respectively.

We first examine the fixed tree structure (Table 3.5). The highest log-likelihood is achieved for $\beta_L = 5$, $\beta_G = 5$, $L = 10$ and $\epsilon_1 = 10^{-4}$. This setting corresponds to small concentration parameters for both the parental (local contribution) and the node (global contribution) priors, therefore giving more importance to the counts coming from the testing piece comparing to other settings of the hyperparameters. In fact, we can observe that in most cases, increasing the value for either concentration parameter leads to decreased performance.

Additionally, performance is always better for the smaller ϵ_1 value (wider tree), which indicates that this model can leverage the noisy information associated with less frequent contexts. On the contrary, in almost all cases performance is

optimal when we use $L = 10$, while including longer contexts leads to marginally worse results, which suggests that the very long contexts stored in the fixed tree structure appear rarely and are not particularly indicative in testing pieces.

Finally, although the model gives better performance when we use small concentration parameters for both priors, we can see that when we increase the contribution from the global model substantially ($\beta_G \geq 50$), then we get better performance if we also increase the contribution from the local model ($\beta_L = 10$).

Table 3.5: Average next-step prediction log-likelihood under the GL_3 combination of the Dirichlet-VMM (mixing the priors) using a fixed tree structure for the local model. Results from 10-fold cross-validation. Boxed: Best overall configuration. Bold: Best configuration for each value of the hyperparameter β_G for the prior distributions in the global model. Performance is optimised when counts from the testing piece are given most importance (small values for both concentration parameters). See text for more details.

		$\epsilon_1 = 10^{-3}$			$\epsilon_1 = 10^{-4}$		
		$\beta_L = 5$	$\beta_L = 10$	$\beta_L = 50$	$\beta_L = 5$	$\beta_L = 10$	$\beta_L = 50$
$\beta_G = 5$	$L = 1$	−1.7360	−1.8005	−2.1140	−1.7342	−1.7990	−2.1134
	$L = 5$	−1.3948	−1.4782	−1.8964	−1.3390	−1.4222	−1.8562
	$L = 10$	−1.3875	−1.4702	−1.8895	**-1.3016**	−1.3756	−1.8129
	$L = 20$	−1.3877	−1.4703	−1.8895	−1.3018	−1.3756	−1.8127
$\beta_G = 10$	$L = 1$	−1.7501	−1.7996	−2.0877	−1.7484	−1.7982	−2.0870
	$L = 5$	−1.4362	−1.4887	−1.8594	−1.3864	−1.4365	−1.8201
	$L = 10$	−1.4301	−1.4815	−1.8526	**-1.3556**	−1.3949	−1.7786
	$L = 20$	−1.4302	−1.4816	−1.8527	−1.3559	−1.3949	−1.7784
$\beta_G = 50$	$L = 1$	−1.8450	−1.8573	−2.0117	−1.8436	−1.8560	−2.0108
	$L = 5$	−1.5950	−1.5910	−1.7599	−1.5624	−1.5512	−1.7245
	$L = 10$	−1.5923	−1.5869	−1.7544	−1.5569	**-1.5299**	−1.6918
	$L = 20$	−1.5925	−1.5870	−1.7544	−1.5573	−1.5300	−1.6917
$\beta_G = 100$	$L = 1$	−1.8942	−1.8986	−1.9940	−1.8930	−1.8973	−1.9930
	$L = 5$	−1.6591	−1.6474	−1.7392	−1.6349	−1.6135	−1.7058
	$L = 10$	−1.6579	−1.6448	−1.7346	−1.6439	**-1.6039**	−1.6788
	$L = 20$	−1.6581	−1.6449	−1.7346	−1.6443	−1.6041	−1.6787
$\beta_G = 200$	$L = 1$	−1.9359	−1.9369	−1.9918	−1.9348	−1.9358	−1.9908
	$L = 5$	−1.7101	−1.6964	−1.7382	−1.6945	**-1.6685**	−1.7067
	$L = 10$	−1.7102	−1.6952	−1.7347	−1.7189	−1.6712	−1.6859
	$L = 20$	−1.7104	−1.6953	−1.7348	−1.7194	−1.6715	−1.6858

When we use a dynamic tree structure for the local model (Table 3.6), that is when we allow the tree to expand according to what we observe in the testing piece, the best performance is achieved with $\beta_L = 5$, $\beta_G = 10$, $L = 20$ and $\epsilon_1 = 10^{-4}$. Similar to the previous case, increasing the concentration parameter for the local prior contribution leads to worse performance, which is particularly evident when look at the results for $\beta_L = 50$. On the contrary, the concentration parameter for the global prior contribution has a smaller effect on performance, although the best setting is again for a small value, $\beta_G = 10$.

Table 3.6: Average next-step prediction log-likelihood under the GL_3 combination of the Dirichlet-VMM (mixing the priors) using a dynamic tree structure for the local model. Results from 10-fold cross-validation. Boxed: Best overall configuration. Bold: Best configuration for each value of the hyperparameter β_G for the prior distributions in the global model. Performance is optimised for small values of the concentration parameters. See text for more details.

		$\epsilon_1 = 10^{-3}$			$\epsilon_1 = 10^{-4}$		
		$\beta_L = 5$	$\beta_L = 10$	$\beta_L = 50$	$\beta_L = 5$	$\beta_L = 10$	$\beta_L = 50$
$\beta_G = 5$	$L = 1$	-1.7343	-1.7992	-2.1135	-1.7342	-1.7990	-2.1134
	$L = 5$	-1.2678	-1.3818	-1.8539	-1.3015	-1.3949	-1.8451
	$L = 10$	-1.1101	-1.2075	-1.7020	-1.1080	-1.2048	-1.6988
	$L = 20$	-1.0976	-1.1850	-1.6716	**-1.0771**	-1.1349	-1.5848
$\beta_G = 10$	$L = 1$	-1.7484	-1.7983	-2.0872	-1.7484	-1.7982	-2.0870
	$L = 5$	-1.2772	-1.3748	-1.8247	-1.3369	-1.4023	-1.8105
	$L = 10$	-1.0975	-1.1877	-1.6765	-1.1010	-1.1877	-1.6714
	$L = 20$	-1.0850	-1.1652	-1.6461	**-1.0685**	-1.1167	-1.5573
$\beta_G = 50$	$L = 1$	-1.8434	-1.8561	-2.0112	-1.8436	-1.8560	-2.0108
	$L = 5$	-1.3420	-1.4074	-1.7416	-1.4776	-1.4907	-1.7166
	$L = 10$	-1.1013	-1.1736	-1.6004	-1.1235	-1.1869	-1.5911
	$L = 20$	-1.0888	-1.1511	-1.5700	**-1.0858**	-1.1118	-1.4764
$\beta_G = 100$	$L = 1$	-1.8926	-1.8973	-1.9935	-1.8930	-1.8973	-1.9930
	$L = 5$	-1.3710	-1.4318	-1.7189	-1.5363	-1.5408	-1.6956
	$L = 10$	-1.1086	-1.1778	-1.5759	-1.1387	-1.1980	-1.5668
	$L = 20$	**-1.0961**	-1.1554	-1.5456	-1.0987	-1.1208	-1.4514
$\beta_G = 200$	$L = 1$	-1.9343	-1.9357	-1.9913	-1.9348	-1.9358	-1.9908
	$L = 5$	-1.3942	-1.4538	-1.7098	-1.5844	-1.5852	-1.6923
	$L = 10$	-1.1154	-1.1834	-1.5620	-1.1524	-1.2098	-1.5543
	$L = 20$	**-1.1029**	-1.1610	-1.5317	-1.1102	-1.1307	-1.4379

Overall, the GL_3 combination performs well for good choices of the concentration parameters. However, for bad choices of the hyper-parameters, and particularly if we use a high concentration parameter for the local prior, performance can decrease substantially. In fact, when the local concentration parameter is too high, the results are worse comparing to the global Dirichlet-VMM, suggesting that the limited information in the local model can result in very noisy estimates.

Table 3.7: Average next step prediction log-likelihood under a local Dirichlet-VMM. The Dirichlet-VMM is trained using only the observations from the test piece. Results from 10-fold cross-validation. Bold: Best configuration.

	$\beta = 5$	$\beta = 10$	$\beta = 50$	$\beta = 100$	$\beta = 200$
$L = 10$	-1.2539	-1.3020	-1.7415	-2.0166	-2.3053
$L = 20$	**-1.2235**	-1.2323	-1.6274	-1.9067	-2.2081

As a final step to evaluate the global versus the local contributions, we run an experiment where we use only a local Dirichlet-VMM for prediction, constructed dynamically for each test sequence. Note that the training corpus is not used in this approach. Results from this experiment are presented in Table 3.7. We can observe that the information coming from the test piece is highly predictive, as the best local model has higher prediction log-likelihood than the best global model. Nevertheless, coupling the two sources of information, using either the GL_1 or the GL_3 combination, outperforms both sources used in isolation. This suggests that the global and local models provide complementary information to the prediction task, with the first one representing overall statistics of the music genre and the latter one specific characteristics of a music piece.

3.4.2 Summary of Findings

We analysed the performance of the VMM, the Dirichlet-VMM and the Global-Local Dirichlet-VMM combinations with respect to different settings of the meta-parameters for the learning algorithm and the hyperparameters of the Dirichlet priors. In all the models performance increases substantially when we move from first-order ($L = 1$) to higher order ($L = \{5, 10\}$) dependencies, but levels off when we further increase the maximum length ($L = 20$). In the VMM pruning the tree according to the frequency counts criterion (ϵ_1) has a considerable effect

on performance, with more pruning leading to better results. The 'additional information' criterion (ϵ_2) and the level of the additive smoothing (γ) have only a marginal effect in performance. For the Dirichlet-VMM performance is optimal when we use all available information (large L/small ϵ_1) and account for the uncertainty through a fairly high prior contribution (large β). For all the tree structures examined the Dirichlet-VMM performs better than the VMM.

Information from the already processed testing piece is highly predictive of the future, with the best setting for any of the global-local combinations giving higher log-likelihood than the best setting of the global Dirichlet-VMM. For the GL_1 coupling scheme longer and wider trees coupled with a moderate prior contribution are best when using a fixed tree structure, while narrower trees with a small concentration parameter are optimal for the dynamic tree structure. In the GL_2 coupling scheme performance is optimised with high values for both concentration parameters and narrower trees, which is expected as the product formulation penalises highly peaked predictive distributions in the experts. Finally, in the GL_3 combination a small concentration parameter for the local (parental) prior is always preferred, indicating that this prior does not contribute much to the prediction task. This is somewhat expected as the local tree is associated with very few datapoints and therefore the estimates for the parent nodes are also noisy.

3.5 Comparative Analysis

In this section we compare model performance. For each model we use the best parametrisation from the cross-validation experiments presented in the previous section. We use two evaluation criteria. The first one is next-step prediction log-likelihood of a held-out, testing set, comprising 64 pieces of music. The second one is the Maximum Mean Discrepancy (MMD) of string kernels computed between the testing pieces and model generations. Note that across all evaluations we use only the first half of each testing sequence to avoid results being dominated by exact repetition.

We report results that are averaged across different folds. More specifically, for each model type we compute the evaluation metrics using the testing data and each

Table 3.8: Average next step prediction log-likelihood of the testing data under different models. For each model type, we use the 10 trained models from cross-validation and compute the average next-step prediction log-likelihood of the testing data (64 pieces) under each of these models. We report the mean and standard deviation across the 10 trained models. For the global models (top) performance increases as we consider more detailed information, with the Dirichlet-VMM.De being best. Incorporating information from the testing piece (bottom) further improves prediction, with the GL_1-dynamic coupling scheme being best. See text for further details.

	Mean $\pm$ St.Dev.	Optimal Setting
Empirical Marginal	-2.2217 ± 0.0009	
VMM	-1.7437 ± 0.0029	$L = 10, \epsilon_1 = 10^{-3}, \epsilon_2 = 20, \gamma = 10^{-2}$
Dirichlet-Bigram	-1.9171 ± 0.0019	$L = 1, \epsilon_1 = 0, \beta = 10$
Dirichlet-VMM.Sh	-1.6084 ± 0.0030	$L = 20, \epsilon_1 = 10^{-3}, \beta = 100$
Dirichlet-VMM.De	-1.5663 ± 0.0040	$L = 20, \epsilon_1 = 10^{-4}, \beta = 100$
GL_1-fixed	-1.4454 ± 0.0038	$L = 20, \epsilon_1 = 10^{-4}, \beta = 50$
GL_1-dynamic	-1.0393 ± 0.0054	$L = 20, \epsilon_1 = 10^{-3}, \beta = 5$
GL_2	-1.4309 ± 0.0052	$L = 20, \epsilon_1 = 10^{-3}, \beta_G = 200, \beta_L = 200$
GL_3-fixed	-1.3072 ± 0.0021	$L = 10, \epsilon_1 = 10^{-4}, \beta_G = 5, \beta_L = 5,$
GL_3-dynamic	-1.0905 ± 0.0012	$L = 20, \epsilon_1 = 10^{-4}, \beta_G = 10, \beta_L = 5,$

of the 10 trained models from the cross-validation experiment above. Therefore, each model is trained using 180 pieces, sampled with replacement from a total of 200 pieces. This allows us to evaluate how each model behaves given different subsamples from the true population.

3.5.1 Next-Step Prediction

Table 3.8 shows the average next step prediction log-likelihood of the testing data under different models. The first line corresponds to the empirical marginal distribution and it is used as a simple baseline. The empirical marginal distribution is the maximum likelihood model if we consider no temporal dependencies, with the probability of each note being proportional to the number of times we have seen this note in the training data. For our comparison we have also included a Dirichlet-Bigram model, where we set the maximum depth of the tree to 1 and do not prune any of the nodes in depth 1.

The relative performance of the models is similar to what we observed in the cross-validation experiment. More specifically, when we only consider the global models, i.e. models that do not consider any information from the test piece, the highest log-likelihood is achieved by the Dirichlet-VMM that considers the most detailed information, denoted by Dir-VMM.De, where 'De' stands for deep. As already mentioned, for the dataset we consider, an ϵ_1 value of 10^{-3} means that a context must be observed roughly 50 times before it is included in the tree and results in shallower and narrower trees (Dirichlet-VMM.Sh), whereas the value of 10^{-4} corresponds to observing a context roughly 5 times (Dirichlet-VMM.De).

The contextual information is very important in this task. We can see that the log-likelihood increases significantly when we move from no temporal dependencies (empirical marginal) to first order dependencies (Dir-Bigram), with a second substantial increment when we consider variable order dependencies with the VMM and the Dirichlet-VMM. The ability of the Dirichlet-VMM to include information coming from the shorter contexts into the predictive probabilities of the longer ones makes it a better model for prediction. It also allows us to take into account noisy information coming from very long or infrequent contexts, which is nevertheless important during prediction.

Looking at the models that combine global with local information, we can see that the GL_1-dynamic combination gives the highest prediction log-likelihood. In this coupling scheme, the posterior distribution over the model parameters is computed using counts from the training data and the already observed test sequence. Additionally, the tree structure of the Dirichlet-VMM changes dynamically to include new contexts that we observe in the test sequence. Including information from the test piece in this manner is very beneficial for the prediction task, as the model is able to capture new informative contexts that are specific to a piece, but also update the statistics of already included contexts, if these are used in a different manner in the test piece. Compared to the GL_1-fixed combination, where we update the counts, but keep a fixed tree structure, we can see that although adapting the predictive distributions of the global contexts increases performance, the most significant improvement, in the context of prediction, comes from allowing new contexts to be included in the tree.

The GL_2 combination, which corresponds to the product of a global and a local expert, performs poorly when compared to the other two global-local combinations,

given the fact that it uses a dynamic tree structure. The main limitation of this model is that the local model is completely independent of the global one and has only limited information, resulting in predictive distributions that are highly peaked. In a product formulation this can be problematic, as we require all of the experts to "like" a configuration in order for it to have high probability.

Comparing the GL_3 (mixing the priors) to the GL_1 (update the counts) combination, we can see that when using a dynamic tree structure the GL_1 combination gives better results. When we allow the tree structure to change, we can accommodate most of the idiosyncrasies of each individual piece and thus we can treat the counts coming from the testing sequence in the same way as counts from the training data. On the other hand, when we have a fixed tree structure, the GL_3 combination gives better results, as it allows us to prioritise the counts coming from the testing data, while still using the information from the global model.

3.5.2 Maximum Mean Discrepancy of String Kernels

In this section we compare model performance by computing the Maximum Mean Discrepancy (MMD) between test sequences and model samples, using a string kernel to quantify the similarity between every pair of sequences. We generate 10 sets of samples from each model type, where each set is sampled from a model trained on a different data fold. We then compute a mismatch kernel between each set of samples and the set of testing sequences. Each sampled set comprises 64 generated sequences having the same length as the 64 testing sequences. We report the mean and the standard deviation of the squared MMD across the 10 sets of samples, together with the minimum and the maximum value attained. We use the $(4, 1)$ mismatch kernel and the normalised $(4, 1)$ mismatch kernel, which are described in Section 2.2.2. Note that preliminary experiments with the $(5, 1)$ and the $(6, 1)$ mismatch kernels gave similar results.

From the models that combine global and local information, described in Section 3.3, we have only included the GL_1-fixed Dirichlet-VMM in this analysis. This is due to the fact that sampling from the other models often results in degenerate samples. Remember that these models combine global information coming from the training data with local information coming from a sequence during testing. However, when we generate samples the local information corresponds

to previously sampled configurations. In coupling schemes where either the local tree construction or the local parameter estimation is based solely on previous generations, sampling often gets stuck to a single short phrase which is assigned higher and higher probability and is perpetually repeated. An example of such a degenerate sample from the GL_1-dynamic Dirichlet-VMM is shown in Figure 3.3. Note that in the next-step prediction task, the global-local combination models outperformed models based on global information only and in particular, models with dynamic tree structures resulted in significantly higher log-likelihood. By examining model samples, it is clear that these are poor generative models, which suggests that the likelihood metric is not always a good indicator of quality.

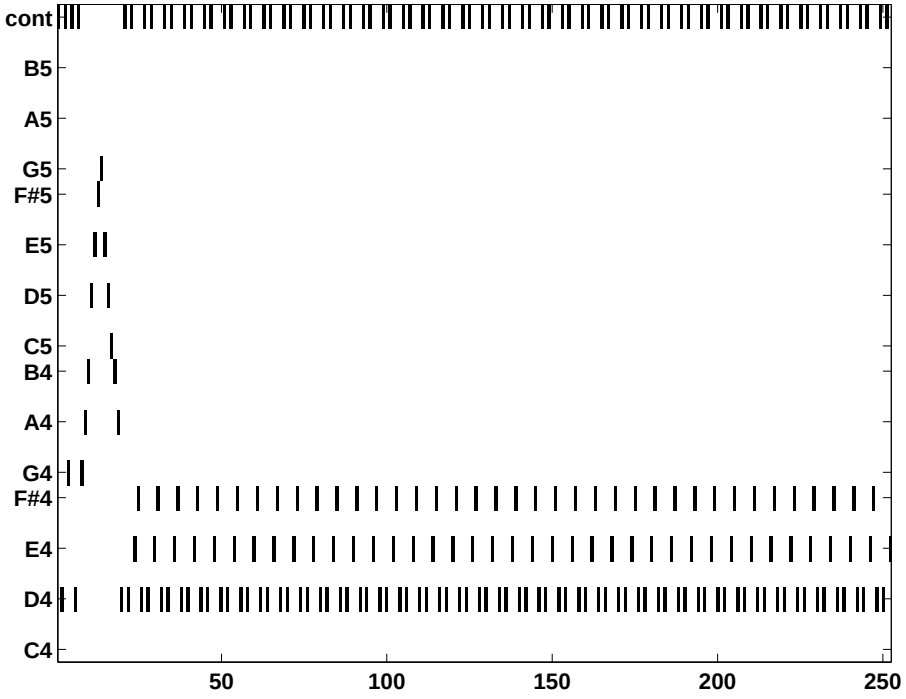


Figure 3.3: Degenerate sample from the GL_1-dynamic Dirichlet-VMM. The tree structure is constructed dynamically during sampling and leads to a single motif being repeated continuously.

Table 3.9 shows the mean and standard deviation of the estimated squared MMD between the testing sequences and the 10 sets of model samples using two different kernels. Again, we use the empirical marginal distribution and the Dirichlet-Bigram as baseline models for this comparative analysis; the former one includes no temporal structure, while the latter one has only first-order Markov dependencies. Additionally, we report the MMD between testing sequences and samples drawn from a Uniform distribution, i.e. a distribution that has no

Table 3.9: Mean, standard deviation, minimum and maximum of the estimated squared MMD between testing sequences and samples generated from different models. Results for the $(4,1)$ and the normalised $(4,1)$ mismatch kernels. Models that consider longer temporal dependencies perform better than models with shorter or no temporal structure. The GL_1-fixed Dirichlet-VMM performs best with either kernel. See text for more details.

	$(4,1)$ Mismatch Kernel			Normalised $(4,1)$ Mismatch Kernel		
	Mean $\pm$ St.Dev.	Min	Max	Mean $\pm$ St.Dev.	Min	Max
Uniform	411.69 ± 1.31	409.34	413.13	0.1657 ± 0.0011	0.1639	0.1668
Emp. Marginal	129.89 ± 10.13	116.04	147.08	0.0644 ± 0.0054	0.0543	0.0723
TrainData	6.22 ± 4.58	1.33	16.71	0.0030 ± 0.0015	0.0002	0.0048
VMM	173.58 ± 8.56	159.81	187.84	0.0269 ± 0.0046	0.0200	0.0360
Dir-Bigram	91.81 ± 6.10	83.67	102.18	0.0373 ± 0.0024	0.0342	0.0407
Dir-VMM.Sh	9.20 ± 4.97	2.71	19.79	0.0257 ± 0.0046	0.0163	0.0342
Dir-VMM.De	10.04 ± 5.23	2.18	18.71	0.0256 ± 0.0034	0.0205	0.0296
GL_1-fixed	8.36 ± 5.40	1.54	21.10	0.0246 ± 0.0040	0.0163	0.0306

information regarding the structure of the data. From table 3.9 we can see that all models greatly outperform the Uniform distribution using either the mismatch kernel or the normalised mismatch kernel.

For the purposes of this analysis, we have also computed the MMD between testing and training sequences. Again, we report the mean and standard deviation of the estimated squared MMD between the 64 testing sequences and 10 sets of 64 training sequences, each set randomly selected from the corresponding fold. This acts as a lower limit as to what can be achieved given that the populations we use to evaluate the MMD have finite size. The testing and training sequences are both drawn from the true theoretical probability distribution, P, that we want to model. Therefore, the MMD between testing and training sequences should converge to 0 as we increase the size of the two populations. Looking at Table 3.9 we can observe that the squared MMD between testing and training sequences is the smallest using either kernel, which indicates that none of the models has learned P completely.

The fact that the MMD between testing and training sequences should converge to 0 reveals a limitation of this evaluation method; a model that replicates the training data exactly would perform well under this measure, without having

Table 3.10: Mean, standard deviation, minimum and maximum of the estimated squared MMD between training sequences and samples generated from different models. Results for the $(4, 1)$ and the normalised $(4, 1)$ mismatch kernels. Results using training sequences are similar to results using testing sequences (Table 3.9). See text for more details.

	$(4,1)$ Mismatch Kernel			Normalised $(4,1)$ Mismatch Kernel		
	Mean $\pm$ St.Dev.	Min	Max	Mean $\pm$ St.Dev.	Min	Max
Emp. Marginal	119.45 ± 26.65	93.15	180.56	0.0792 ± 0.0067	0.0694	0.0924
VMM	121.36 ± 30.56	86.31	185.80	0.0171 ± 0.0066	0.0092	0.0277
Dir-Bigram	71.40 ± 21.65	50.30	115.02	0.0453 ± 0.0051	0.0390	0.0538
Dir-VMM.Sh	6.72 ± 7.01	0.84	23.37	0.0370 ± 0.0084	0.0194	0.0497
Dir-VMM.De	8.83 ± 7.82	0.98	24.33	0.0372 ± 0.0107	0.0234	0.0580
GL_1-fixed	6.09 ± 8.35	0.88	27.89	0.0340 ± 0.0101	0.0170	0.0525

learned anything about the underlying structure. For this reason, we also present results where we compare the sampled generations with training sequences, to examine whether replication is occurring. To perform this analysis we randomly sample a set of 64 pieces from the training data in each fold and compare them against the 64 sampled pieces of the same fold. Results from this evaluation are shown in Table 3.10.

To facilitate the analysis of our findings, the results from Tables 3.9 and 3.10 are also shown in Figure 3.4, where we have excluded the Uniform distribution, as it dominates the remaining results, making it difficult to visualise the differences among the rest of the models.

Looking at the barplots with the squared MMD between testing sequences and models samples (top), we can see that the Dirichlet-VMM models perform the best with either kernel. Comparing the two subplots (top, left and right), we can see that in some cases the relative performance of a model changes when we use the non-normalised and the normalised mismatch kernel. This is particularly evident for the Variable-Length Markov Model (VMM), which has the worst performance when we use the non-normalised kernel, while with the normalised kernel it outperforms the empirical marginal and the Dirichlet-Bigram and is comparable to the Dirichlet-VMM models. Additionally, the MMD for the Dirichlet-VMMs is very close to the one for the training data with the non-normalised kernel, but it is considerably higher when we use the normalised kernel. These observations reveal

a second limitation of this evaluation method, namely that the result depends strongly on the choice of the kernel. We will examine this further by visualising the kernel matrices in the next section.

The squared MMD between training sequences and model samples (bottom two figures) is very similar to the one between the testing sequences and model samples. In the case of the non-normalised kernel (left) we can see that the squared MMD for all the models is, in fact, moderately lower when we consider training instead of testing data, with the VMM having the highest difference between the two. Nonetheless, the squared MMD between training sequences and model samples is still higher than the one between training and testing sequences, thus making exact replication less likely. With the normalised kernel, the squared MMD for all models is higher when we consider the training data than when we use the testing data, with the exception of the VMM. In the next section we examine aspects of the normalised kernel that can explain this behaviour. Finally, we can observe that the standard deviation across folds is higher when we use the training data. This is due to the experimental procedure. In the top barplots the MMD is computed using 10 sets of different samples, but only 1 set of testing data. On the contrary, in the bottom barplots, the MMD is computed using the 10 sets of different samples as before, each time compared against a different set of training data. Therefore, the standard deviations in the bottom plots contain a variance component that arises due to the variation of the data sequences, which is absent in the former case.

Overall, we can say that under this evaluation metric, models that consider longer temporal dependencies tend to perform better than models with shorter or no temporal structure. However, we can note that the performance of the Dirichlet-VMM is almost the same regardless of whether we use the deeper ($.De$) or the shallower ($.Sh$) tree structure. In fact, for the non-normalised kernel, the model with the deeper tree performs slightly worse. Additionally, we can see that the best performing model in both cases is the GL_1-fixed Dirichlet-VMM, where global information from the training data is combined with local information coming from the already generated sequence. Although the improvement to using a global Dirichlet-VMM is small, it suggests that allowing some aspect of the model to be uniquely defined for each sequence can be beneficial when modelling music. We will examine this more in the next section.

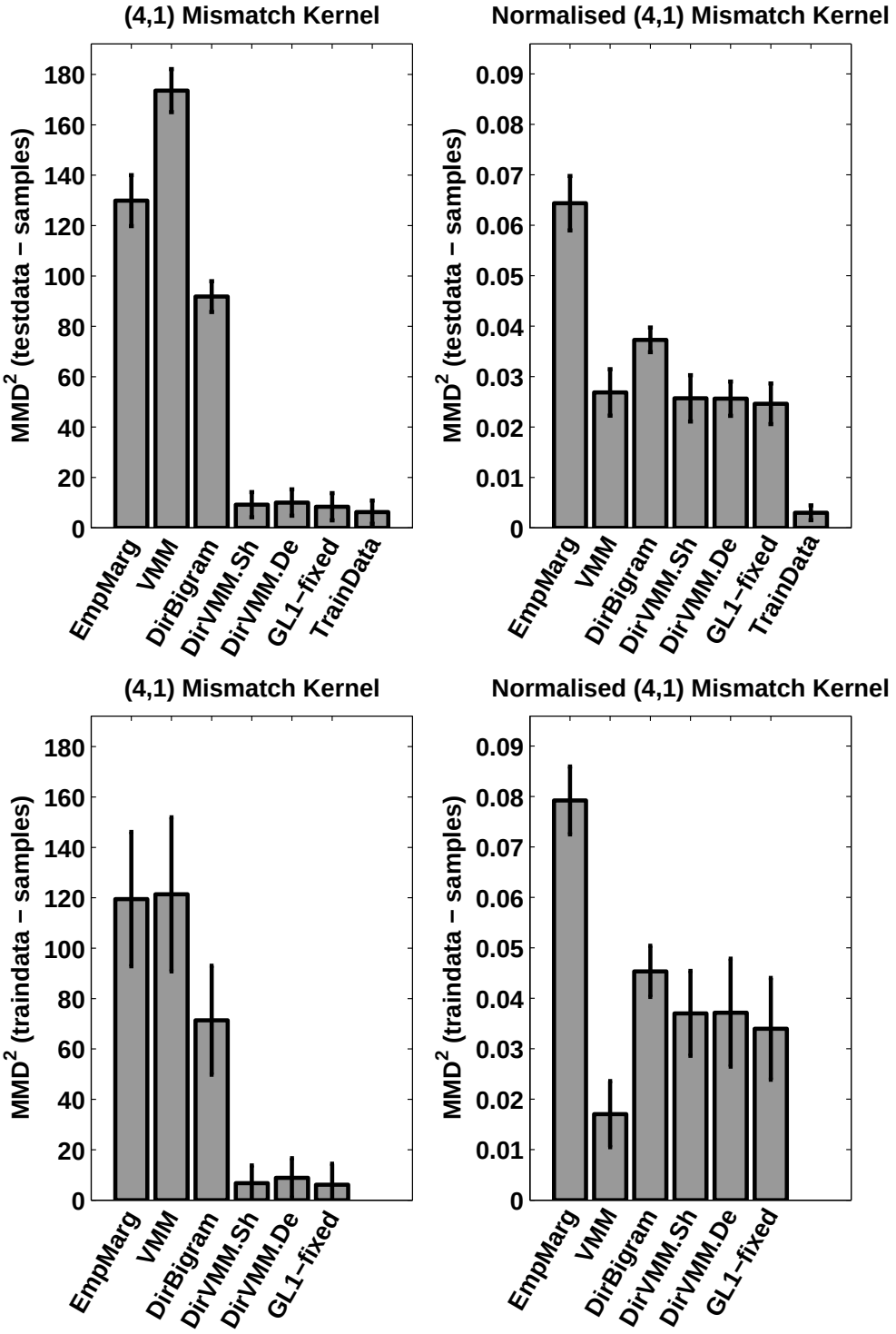


Figure 3.4: Barplots depicting the mean and standard deviation of the estimated squared MMD between data sequences and sequences sampled from different models. Top: squared MMD between testing sequences and model samples. Bottom: squared MMD between training sequences and model samples. Left: results using the $(4, 1)$ mismatch kernel. Right: results using the normalised $(4, 1)$ mismatch kernel. See text for details.

3.5.2.1 Visualising the Kernel Matrices

In order to better understand the results from the previous section, we examine the kernel matrices used to estimate the MMD. Each kernel matrix is symmetric with K_{ij} being the similarity between sequence i and sequence j and K_{ii} being the similarity of sequence i to itself. The first 64 rows (and columns) of each matrix correspond to testing sequences and the following 64 rows (and columns) correspond to model samples. We present the kernel matrices computed using samples from fold 1, but note that the other folds give similar results. Figure 3.5 shows the $(4, 1)$ mismatch kernel between testing sequences and samples from different models. We can observe that samples drawn from the Uniform distribution share very few subsequences with testing sequences, but also with each other. This is also the case for samples drawn from the Variable-Length Markov Model, although in this case we begin to observe some structure in the test-sample and sample-sample parts of the kernel.

In order to visualise the structure of the kernels in more detail, we truncate very high values to $1,000$, so that the colormap is not dominated by a few highly repetitive sequences. Note that roughly 5% of the elements of each kernel matrix are truncated. Figure 3.6 presents the truncated kernel matrices, where the above observation regarding the Uniform distribution and the VMM is more apparent. Furthermore, we can notice that the test-test part of the kernel (upper left square) has a different structure to the sample-sample part (lower right square) for all the models. The values in the test-test part tend to be more heterogeneous, with some pieces being very similar to each other and some pieces being very different from each other. On the other hand, the generated samples tend to have a more uniform level of similarity, although in the Dirichlet-VMM models we begin to have some differentiation.

Looking at the kernel matrices for the Dirichlet-VMM models we can see that samples from these models have higher similarity both with each other and with the true pieces. However, again the level of similarity in the sample-sample part of the kernel tends to be more uniform than the test-test part, with most pairs of samples being closer to each other than two dissimilar data sequences, but not as close as two highly similar data sequences.

On the other hand samples from the VMM are not very similar to each other,

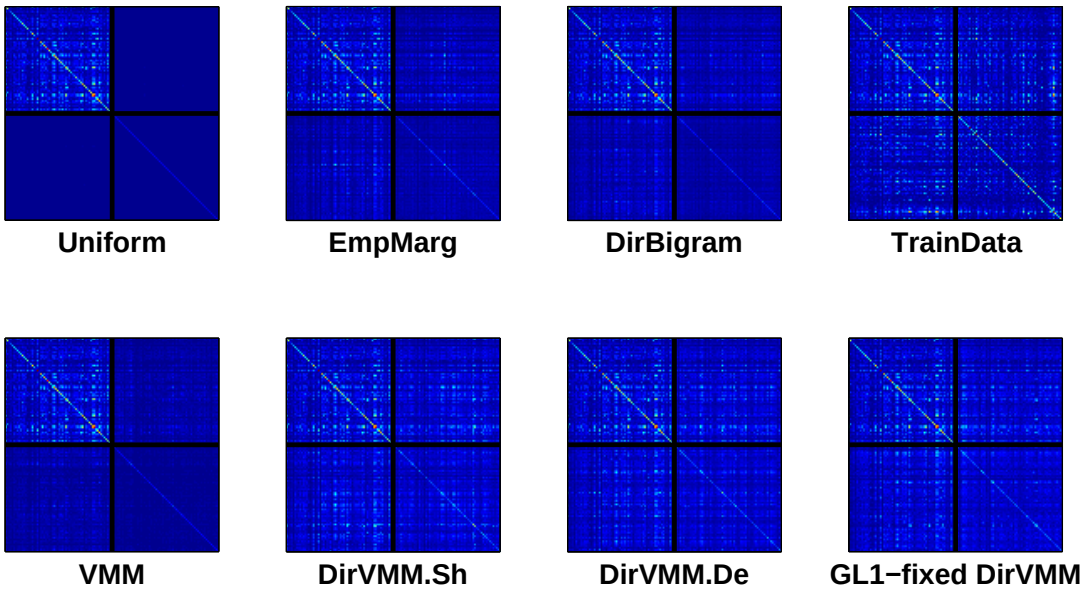


Figure 3.5: The $(4, 1)$ mismatch kernel matrices between testing sequences (rows:1-64) and model samples (rows:65-128) for different models. Colour values range from 0 (dark blue–smallest element across all matrices) to $5,617$ (dark red). Samples from the Uniform distribution have low similarity with each other and with the data sequences. As we move to more expressive models we begin to observe some structure in the data-sample and sample-sample part of the kernel matrix. See text for more details.

although they exhibit some similarity to the testing sequences. The additive smoothing applied in the VMM assigns a lot of the probability mass to events that have never been observed. As a result, the space of possible configurations when sampling from the VMM can be very different to the space of true pieces. Samples from this model are less constrained to the statistics of the data, leading to samples that have lower level of similarity with each other.

Figure 3.7 shows the normalised $(4, 1)$ mismatch kernel for different models. The first thing we can observe is that in the normalised kernels for the Dirichlet-VMM models, the sample-sample part has a higher level of similarity than the test-test part. This is due to the fact that these two populations have different inter and intra-sequence structure. More specifically, looking back at Figures 3.5 and 3.6 we can see that diagonal entries have higher values than off-diagonal entries and the difference between diagonal and off-diagonal entries is higher in the data-data part of a kernel. A musical piece has repeated themes not present in other pieces, making the mismatch kernel measurement especially high when comparing a piece

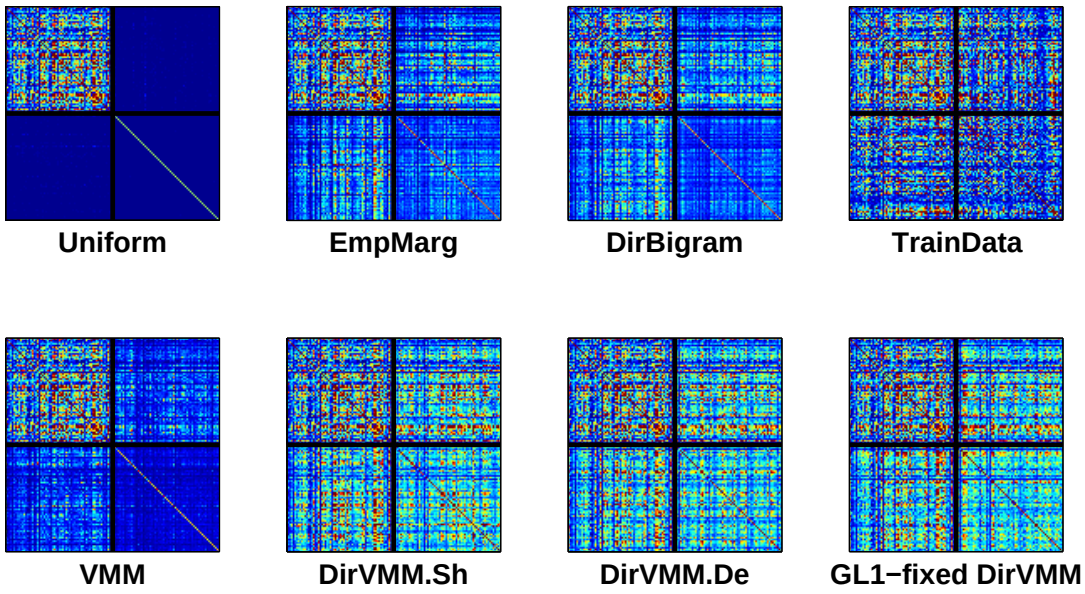


Figure 3.6: The $(4, 1)$ mismatch kernel matrices between testing sequences (rows:1-64) and model samples (rows:65-128) for different models. Colour values range from 0 (dark blue–smallest element across all matrices) to $1,000$ (dark red). Values larger than $1,000$ are truncated. Samples from the Dirichlet-VMMs are more similar to data sequences than samples from the rest of the models, but the sample-sample part of the matrices is more 'uniform' than the data-data part. See text for more details.

to itself (diagonal elements). Put differently, a subsequence from a data piece is more likely to be similar to a another subsequence from the same piece, than to a subsequence from a different piece. On the contrary, a subsequence from a Dirichlet-VMM sample will be as similar to another subsequence from the same sample as to a subsequence from a different sample, as the Dirichlet-VMM does not explicitly model repetition of themes within a single sequence. The GL_1-fixed Dirichlet-VMM combination allows for some differentiation between samples, as it pushes the generated sequence to re-sample the same configurations. However, the model is not very flexible and thus the improvement is small.

Using the normalised mismatch kernel in the MMD evaluation has two benefits compared to using the non-normalised mismatch kernel. Firstly, with the normalised mismatch kernel, we take into account both the inter and the intra-sequence structure. Remember that the estimator for the squared MMD (Equation (2.7)) considers only the off-diagonal entries of a kernel matrix. In the mismatch kernel, the off-diagonal entries are similarities between different sequences, and

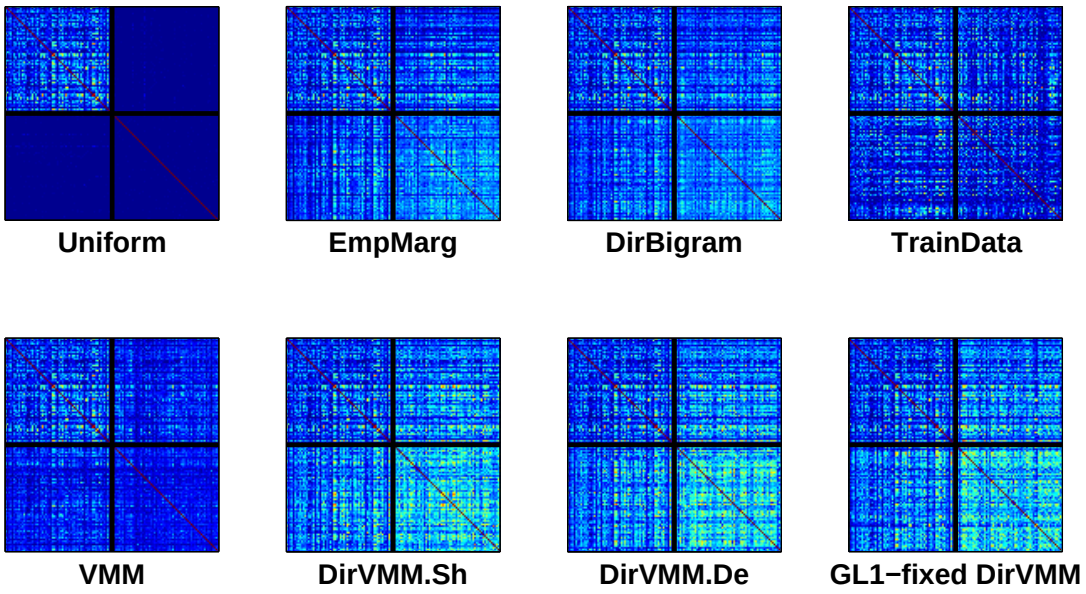


Figure 3.7: The normalised $(4, 1)$ mismatch kernel matrices between testing sequences (rows:1-64) and model samples (rows:65-128) for different models. The models cannot distinguish between inter and intra-sequence statistics. This is reflected in the Dirichlet-VMMs, where the sample-sample part of the matrices have a higher level of similarity than the data-data part. See text for more details

thus the estimated squared MMD expresses how close are model samples to data sequences based on inter-sequence structure. In the normalised mismatch kernel, the similarity between two different sequences (off-diagonal entries) is weighted by the similarity of each of these sequences to itself (diagonal entries). Therefore, the estimated squared MMD depends on both the inter-sequence and the intra-sequence structure. Secondly, the normalised mismatch kernel can smooth out the effect of highly repetitive sequences, which could dominate our computation of the MMD.

However, coupling the contribution from the inter and the intra-sequence structure through normalisation can also be problematic, as in the case of the Variable-Length Markov Model (VMM). Due to the very low intra-sample similarity in the VMM, the squared MMD using the normalised kernel is relatively low, which is counter-intuitive given that samples from the VMM also exhibit low similarity to the data pieces. Given that the results of this evaluation are subject to the selected kernel and its appropriateness for the task in hand, it is advisable to examine the resulting kernel matrices to identify potential limitations similar to

the one presented above.

Note that the similarity matrices computed between model samples and training data have very similar structure to the ones presented above.

3.6 Discussion

In this Chapter we reviewed the variable-length Markov model, which is considered to be state of the art in automatic melody generation. We then developed the Dirichlet variable-length Markov model, which uses iteratively refined Bayesian agents to leverage all available information, while keeping the inference procedure analytically tractable. Under the Dirichlet-VMM formulation smoothing is performed in an implicit manner that uses local Bayesian agents to maintain information coming from the shorter contexts into the predictive probabilities of the longer ones. The Dirichlet-VMM can also be understood as performing Jelinek-Mercer (linear interpolation) smoothing in the VMM, with the weights for the higher and lower order models being automatically acquired through Bayesian inference.

Using two different evaluation metrics we show that the Dirichlet-VMM performs better than the VMM with additive smoothing. Both evaluation methods suggest that contextual information is predictive in melody. Additionally, both evaluation methods reveal that novelty and repetition are important aspects of musical structure. In the next-step prediction task, we see that using information coming from the testing piece significantly improves performance, and the more local information we use the better we typically get. We note, however, that the models with the highest log-likelihood – namely the GL_1-dynamic Dirichlet-VMM and the GL_3-dynamic Dirichlet-VMM – are poor generative models. In these models the local parameters are estimated using only previous time-steps in a sequence. This is beneficial in the context of next-step prediction, as the local tree captures the structure of the already seen testing piece. But when we sample new sequences, this is problematic, as it often reinforces a single motif and leads to degenerate samples.

In the MMD evaluation we again see that the stationarity assumption of the Dirichlet-VMM is not appropriate for melody generation, as true music pieces

exhibit different inter and intra-sequence structure, with each piece typically having repeated themes not present in other pieces. Samples from the GL_1-fixed Dirichlet-VMM have moderately different inter and intra-sequence similarities, as model parameters are updated dynamically while each sample is generated, which pushes the model to re-sample the same configurations within a single sequence. Allowing the structure of the tree to change for each piece would be a more flexible way of allowing for differentiation, but it needs to be guided so that it does not get stuck to degenerate modes. One possibility would be to use the first half of a testing sequence to construct the local tree structure and use this structure to sample a new realisation for the second half of the piece. In Chapter 5 we will examine how we can overcome the stationarity assumption of the Dirichlet-VMM by coupling it with the topic modelling formalism.

A final point to consider is our model selection procedure. As described in Section 2.3.2, hyper-parameter optimisation is done by maximising the data log-likelihood during cross-validation. However, as described above, the log-likelihood and MMD evaluations give opposite results in the case of the GL_1-dynamic and the GL_3-dynamic Dirichlet-VMMs. This means that log-likelihood is not always a good proxy of performance for the MMD evaluation, and therefore optimising hyper-parameters with respect to the MMD objective during cross-validation should be preferred in future work.

Chapter 4

Unsupervised Latent Feature Extractors

In this Chapter we introduce the Time-Convolutional Restricted Boltzmann Machine (TC-RBM) for the task of melody modelling. Models based on the Restricted Boltzmann Machine (RBM) have been successful in a variety of different domains, including static data, such as hand-written digits (Hinton et al., 2006), natural images (Lee et al., 2009), movie ratings (Salakhutdinov et al., 2007) and text documents represented as bags of words (Salakhutdinov and Hinton, 2009), but also sequential data, such as human motion (Taylor et al., 2007; Taylor and Hinton, 2009).

One of the main advantages of the Restricted Boltzmann Machine is its ability to extract good latent features in an unsupervised manner. More specifically, the RBM learns a distributed representation of the input space by discovering the different sources of variation in the data and decoupling them into a set of latent features that can be "on" or "off". This way, the input space is effectively mapped to an exponentially large set of configurations corresponding to the different combinations of active features.

Componential influences are intrinsic to music, too. More specifically, although pieces from the same genre are built using the same musical form, i.e. the same structural principles, the statistical relations among and within the melodies of different pieces are highly complex, as melody depends on several different components, such as the scale, the rhythm and the meter, which in many cases

interdepend on each other. Therefore, we believe that models based on the RBM can be very beneficial in this context, as they are able to identify the different factors causing variation in melody and decompose them into a set of latent musical features.

4.1 The Restricted Boltzmann Machine

The Restricted Boltzmann Machine (RBM) is an undirected, graphical model that consists of a visible and a hidden layer of binary, logistic units. It is a special, bipartite form of the Boltzmann Machine (Ackley et al. (1985)), in which the interaction terms are restricted to units from different layers. Therefore, an RBM forms an undirected network that is fully-connected between layers, but has no visible-to-visible or hidden-to-hidden connections. An RBM with 4 visible and 3 hidden units is depicted in Figure 4.1.

In the RBM, the joint distribution over observed and latent variables is defined through an energy function, which associates a scalar energy to each configuration the variables can take. Let $\mathbf{x}$ be a set of observed variables, $\mathbf{h}$ a set of hidden variables and $\boldsymbol{\theta}$ a vector denoting model parameters. Then an energy-based probabilistic model defines the joint distribution of $\mathbf{x}$ and $\mathbf{h}$ as

$$P(\mathbf{x}, \mathbf{h}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(-E(\mathbf{x}, \mathbf{h}; \boldsymbol{\theta})\right) \quad , \tag{4.1}$$

where $E(\cdot)$ denotes the energy of a configuration and $Z(\boldsymbol{\theta})$ is the partition function, a normalising constant that sums over all possible configurations

$$Z(\boldsymbol{\theta}) = \sum_{\mathbf{x},\mathbf{h}} \exp\left(-E(\mathbf{x}, \mathbf{h}; \boldsymbol{\theta})\right) \quad . \tag{4.2}$$

In the commonly studied case, the RBM has binary, logistic units and its energy function is defined as

$$E(\mathbf{x}, \mathbf{h}|\boldsymbol{\theta}) = -\mathbf{c}^{\mathsf{T}}\mathbf{x} - \mathbf{b}^{\mathsf{T}}\mathbf{h} - \mathbf{x}^{\mathsf{T}}\mathbf{W}\mathbf{h} \quad , \tag{4.3}$$

where $\mathbf{c}$ and $\mathbf{b}$ are bias terms for the visible and hidden units respectively, and $\mathbf{W}$ is a weight matrix for the interaction terms.

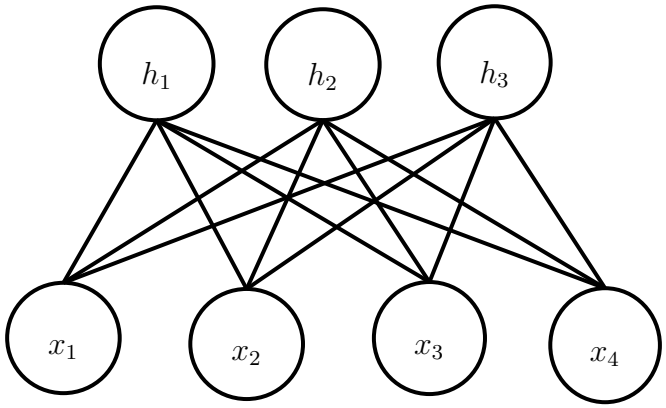


Figure 4.1: A Restricted Boltzmann Machine with 4 visible and 3 hidden units.

4.1.1 Inference and Learning

From Figure 4.1 we can see that due to the absence of intra-layer connections, the hidden units are conditionally independent given the visible ones and vice versa. This is also apparent from the bilinear form of the energy function in (4.3). This makes inference in the RBM easy, as the conditional distribution of the hidden units given the data factorises completely. More specifically, we have

$$P(\mathbf{h}|\mathbf{x}, \boldsymbol{\theta}) = \prod_j P(h_j|\mathbf{x}, \boldsymbol{\theta}) \quad \text{and} \tag{4.4a}$$

$$P(\mathbf{x}|\mathbf{h}, \boldsymbol{\theta}) = \prod_i P(x_i|\mathbf{h}, \boldsymbol{\theta}) \ . \tag{4.4b}$$

For binary logistic units and the energy function in (4.3) we have

$$P(h_j = 1|\mathbf{x}, \boldsymbol{\theta}) = \sigma(b_j + \mathbf{x}^\mathsf{T} W_{\cdot,j}) \quad \text{and} \tag{4.5a}$$

$$P(x_i = 1|\mathbf{h}, \boldsymbol{\theta}) = \sigma(c_i + W_{i,\cdot}^\mathsf{T} \mathbf{h}) \ , \tag{4.5b}$$

where $\sigma(z) = 1/(1 + \exp(-z))$ is the sigmoid function and $W_{i,\cdot}$ and $W_{\cdot,j}$ are the i-th row and the j-th column of the weight matrix $\mathbf{W}$, respectively.

Note that the units need not be binary. We refer the reader to Welling et al. (2005) for a general formulation, where the conditional distributions for the visible and the hidden units can be any member of the exponential family.

Learning in the RBM can be performed in an unsupervised manner, where we effectively allow the model to modify its energy function so that it assigns high energy to improbable configurations and low energy to the desired ones. The

likelihood of a dataset D, consisting of N observations, $D = \left\{\mathbf{x}^1, \ldots, \mathbf{x}^N\right\}$, under the RBM is given by

$$P(D|\boldsymbol{\theta}) = \prod_{n=1}^{N} P(\mathbf{x}^n|\boldsymbol{\theta}) = \prod_{n=1}^{N} \sum_{\mathbf{h}^n} P(\mathbf{x}^n, \mathbf{h}^n|\boldsymbol{\theta}) \ . \tag{4.6}$$

It can be shown that the gradient of the data log-likelihood with respect to the parameters $\boldsymbol{\theta}$ can be written as (see for example Bengio (2007) for a derivation)

$$\frac{\partial \log P(D|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\left\langle \frac{\partial E(\mathbf{x}, \mathbf{h})}{\partial \boldsymbol{\theta}} \right\rangle_0 + \left\langle \frac{\partial E(\mathbf{x}, \mathbf{h})}{\partial \boldsymbol{\theta}} \right\rangle_\infty \tag{4.7}$$

$$\text{where} \quad \begin{cases} \left\langle \frac{\partial E(\mathbf{x},\mathbf{h})}{\partial \boldsymbol{\theta}} \right\rangle_0 = \sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{x}) \frac{\partial E(\mathbf{x}, \mathbf{h})}{\partial \boldsymbol{\theta}} \\ \left\langle \frac{\partial E(\mathbf{x},\mathbf{h})}{\partial \boldsymbol{\theta}} \right\rangle_\infty = \sum_{\mathbf{x},\mathbf{h}} P(\mathbf{x}, \mathbf{h}) \frac{\partial E(\mathbf{x}, \mathbf{h})}{\partial \boldsymbol{\theta}} \end{cases}$$

The first term is an average with respect to the data distribution, while the second one is an average with respect to the model distribution. Note that in Boltzmann Machines and Restricted Boltzmann Machines $\frac{\partial E(\mathbf{x},\mathbf{h})}{\partial \boldsymbol{\theta}}$ is easy to compute. Therefore, maximum likelihood learning can be performed if there is an efficient way to get samples from the two distributions. In an RBM the expectation with respect to the data distribution can be computed easily due to the factorisation in (4.4a). However, to get samples from the model distribution we need to construct a Markov Chain Monte Carlo, for instance through block Gibbs sampling, where we iteratively sample the hidden units given the visibles and the visibles given the hidden. Using Equations (4.4a) and (4.4b), and an initial configuration for the visible vector, $\mathbf{x}_0$, the Markov chain is constructed as

$$\begin{aligned} \mathbf{h}_0 &\sim P(\mathbf{h}|\mathbf{x}_0, \boldsymbol{\theta}) \\ \mathbf{x}_1 &\sim P(\mathbf{x}|\mathbf{h}_0, \boldsymbol{\theta}) \\ \mathbf{h}_1 &\sim P(\mathbf{h}|\mathbf{x}_1, \boldsymbol{\theta}) \\ &\vdots \\ \mathbf{x}_k &\sim P(\mathbf{x}|\mathbf{h}_{k-1}, \boldsymbol{\theta}) \\ \mathbf{h}_k &\sim P(\mathbf{h}|\mathbf{x}_k, \boldsymbol{\theta}) \ , \end{aligned}$$

where k indexes full steps of the chain. As $k \to \infty$, the MCMC converges to the model distribution from which we can get an unbiased sample for estimating the second term in (4.7).

Running the MCMC chain to convergence after each update of the parameters is very expensive. Therefore, parameter estimation in Restricted Boltzmann Machines is often performed by more efficient procedures, which typically approximate the maximum likelihood objective. One such approach that has been widely used in the RBM literature is the Contrastive Divergence (CD) learning rule (Hinton, 2002). In CD-k we initialise the Markov chain using a data vector and collect a sample after running only k steps of the chain, instead of running the chain to equilibrium. This introduces bias to the gradient computation, but has been shown to work well in practice, sometimes even for $k = 1$. In CD-k the parameters are updated according to

$$\Delta\boldsymbol{\theta} = \epsilon \left(- \left\langle \frac{\partial E(\mathbf{x}, \mathbf{h})}{\partial \boldsymbol{\theta}} \right\rangle_0 + \left\langle \frac{\partial E(\mathbf{x}, \mathbf{h})}{\partial \boldsymbol{\theta}} \right\rangle_k \right) \ , \tag{4.8}$$

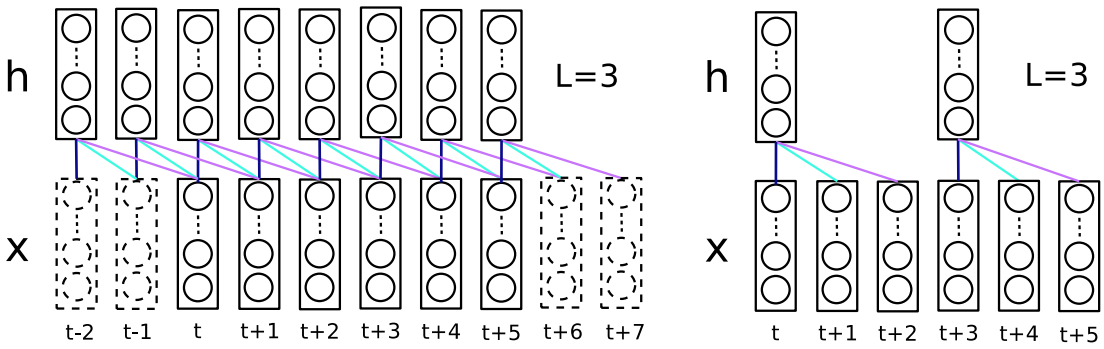
where ϵ is the learning rate.

4.2 The Time-Convolutional Restricted Boltzmann Machine

In this Section we introduce the Time Convolutional Restricted Boltzmann Machine (TC-RBM) (Spiliopoulou and Storkey, 2011) as a new way of modelling sequential data with an RBM type network. The TC-RBM is an adaptation of the Convolutional RBM for sequences and it is motivated by the successful application of such models in static image data (Lee et al., 2009; Norouzi et al., 2009).

Previous RBM approaches to sequence modelling use the RBM to model a single time-step and attempt to capture the temporal relations in the data by introducing different types of directed connections from units in previous time-steps (Taylor et al., 2007; Sutskever and Hinton, 2007; Taylor and Hinton, 2009). By contrast, the TC-RBM is a fully undirected network and attempts to capture the structure of music at a motif (subsequence) level rather than a single time-step.

Figure 4.2 depicts a TC-RBM (left) and a RBM modelling subsequences (right). In the following description we refer to the latter model as seqRBM. In both models the hidden units receive input from visible units in a subsequence of length L. By considering visible subsequences – rather than single time-steps – the



(a) TCRBM (b) seqRBM

Figure 4.2: (a) A Time-Convolutional RBM with filter size $L = 3$. (b) A RBM modelling subsequences of length $L = 3$. Connections with the same colour correspond to the same weight parameters. In both models the hidden units receive input from all visible units in a subsequence of length L. In the TC-RBM the hidden units are shared in time, so that a visible unit at time t receives input from hidden units in time-steps $t - L + 1$ to t (see text for more details). By contrast, in the RBM presented in (b) each visible subsequence is independent from other visible subsequences.

models are able to capture local temporal dependencies in the data. The hidden units learn valid configurations over whole subsequences and can thus capture frequent motifs and their transformations. The difference between the seqRBM and the TC-RBM is that in the former each visible subsequence is independent from other visible subsequences, whereas in the latter the network connectivity induces dependencies between all visible time-steps in a sequence.

To understand this better, we can consider the connections of the visible units in the two models. In the seqRBM, the visible units at time-steps t to $t + 2$ are all connected to the same hidden units. Similarly, the visible units at time-steps $t + 3$ to $t + 5$ are all connected to the same hidden units, which are different to the hidden units for the visible subsequence in t to $t + 2$. Hence, the seqRBM corresponds to a standard RBM (Equation (4.1)), where $\mathbf{x}$ is the visible vector we acquire by concatenating three subsequent time-steps.

In the TC-RBM, the visible units at time-step t are connected to hidden units in time-steps $t - 2$ to t, which are in turn connected to visible units spanning time-steps $t - 2$ to $t + 2$. Similarly, the visible units at time-step $t + 2$ are connected to hidden units in time-steps t to $t + 2$, which are in turn connected to visible units spanning time-steps t to $t + 4$. Therefore, in the TC-RBM the hidden units

propagate information through time and induce dependencies between visible units spanning the entire visible sequence.

Additionally, the connectivity of the TC-RBM removes the need to partition a sequence into subsequences and thus the need to explicitly deal with boundary effects at the subsequence level. By contrast, in the seqRBM, the fact that each visible subsequence is independent of the following one, introduces discontinuities in the edges (boundaries) of the subsequences, which we need to smooth out if we are interested in modelling whole sequences.

However, in the TC-RBM we need to deal with the boundaries of a sequence. As illustrated in Figure 4.2 by the dashed plates, in order to infer the units at the beginning and ending of the hidden sequence, we need to pad the beginning and ending of the visible sequence with $L-1$ vectors. In the experiments presented here, we use the zero vector for padding.

The energy function in the TC-RBM is defined over a whole sequence, $\mathbf{x}_{1:T}$, but can be decomposed over time-steps. Let $\mathbf{x}_{1:T}$ be a visible sequence and $\mathbf{h}_{1:T}$ be the corresponding latent representation. Then, the energy function of the TC-RBM can be written as

$$E\left(\mathbf{x}_{1:T}, \mathbf{h}_{1:T} | \boldsymbol{\theta}\right) = -\sum_{t=1}^{T}\left(\mathbf{c}^{\mathsf{T}}\mathbf{x}_t + \mathbf{b}^{\mathsf{T}}\mathbf{h}_t + \sum_{\ell=1}^{L}\mathbf{x}_{t+\ell-1}^{\mathsf{T}}\mathbf{W}_\ell\mathbf{h}_t\right), \tag{4.9}$$

where $\mathbf{x}_{1:T}$ is a visible sequence, $\mathbf{h}_{1:T}$ is the corresponding hidden configuration and L is the size of the filter we apply, i.e. the number of visible time-steps that a hidden unit receives input from. The interaction terms are parametrised by a 3-dimensional array $\mathbf{W}$, where each slice ℓ of the array is the weight matrix for the connections of hidden units at time t with visible units at time $t+\ell-1$ and $\ell \in \{1, \cdots, L\}$. The biases $\mathbf{c}$ and $\mathbf{b}$, for the visible and hidden units respectively, are the same for all time-steps.

In principle, we could have different weights (and biases) for each time-step. However, learning such a model would require an enormous amount of data, and so in the TC-RBM the weights (and biases) are constrained to be the same across time-steps. In Figure 4.2, weights constrained to the same value are depicted using connections of the same colour. Due to this weight sharing, inference in the TC-RBM can be performed efficiently by convolving (appropriately shifted) slices of the weight array with the visible and the hidden sequence along the time

dimension. The operations we need to perform are sliding dot products along time and we use the fast Fourier transform (FFT) to accelerate their computation. This procedure is described in Section 4.2.1.

Similarly to an RBM, the joint probability distribution of an observed sequence and the corresponding latent configuration under the TC-RBM is defined as

$$P(\mathbf{x}_{1:T}, \mathbf{h}_{1:T}|\boldsymbol{\theta}) = \frac{\exp\left(-E\left(\mathbf{x}_{1:T}, \mathbf{h}_{1:T}|\boldsymbol{\theta}\right)\right)}{Z(\boldsymbol{\theta})} \ . \tag{4.10}$$

The conditional probability distributions of this model factorise over time-steps and units. We have

$$\begin{aligned}
P(\mathbf{h}_{1:T}|\mathbf{x}_{1:T}, \boldsymbol{\theta}) &= \frac{P(\mathbf{x}_{1:T}, \mathbf{h}_{1:T}|\boldsymbol{\theta})}{\sum_{\tilde{\mathbf{h}}_{1:T}} P(\mathbf{x}_{1:T}, \tilde{\mathbf{h}}_{1:T}|\boldsymbol{\theta})} \\
&= \frac{\exp(-E(\mathbf{x}_{1:T}, \mathbf{h}_{1:T}|\boldsymbol{\theta}))}{\sum_{\tilde{\mathbf{h}}_{1:T}} \exp(-E(\mathbf{x}_{1:T}, \tilde{\mathbf{h}}_{1:T}|\boldsymbol{\theta}))} \\
&= \frac{\prod_{t=1}^{T} \exp(\mathbf{c}^\mathsf{T}\mathbf{x}_t + \mathbf{b}^\mathsf{T}\mathbf{h}_t + \sum_{\ell=1}^{L} \mathbf{x}_{t+\ell-1}^\mathsf{T}\mathbf{W}_\ell\mathbf{h}_t)}{\sum_{\tilde{\mathbf{h}}_{1:T}} \prod_{t=1}^{T} \exp(\mathbf{c}^\mathsf{T}\mathbf{x}_t + \mathbf{b}^\mathsf{T}\tilde{\mathbf{h}}_t + \sum_{\ell=1}^{L} \mathbf{x}_{t+\ell-1}^\mathsf{T}\mathbf{W}_\ell\tilde{\mathbf{h}}_t)} \\
&= \frac{\prod_{t=1}^{T} f_t(\mathbf{x}_{1:T}, \boldsymbol{\theta}) \prod_j g_t(\mathbf{x}_{1:T}, h_{j,t}, \boldsymbol{\theta})}{\sum_{\tilde{\mathbf{h}}_{1:T}} \prod_{t=1}^{T} f_t(\mathbf{x}_{1:T}, \boldsymbol{\theta}) \prod_j g_t(\mathbf{x}_{1:T}, \tilde{h}_{j,t}, \boldsymbol{\theta})} \ ,
\end{aligned} \tag{4.11}$$

where $f_t(\mathbf{x}_t) = \exp(\mathbf{c}^\mathsf{T}\mathbf{x}_t)$ and $g_t(\mathbf{x}_{1:T}, h_{j,t}, \boldsymbol{\theta}) = \exp(b_j h_{j,t} + \sum_{\ell=1}^{L} \mathbf{x}_{t+\ell-1}^\mathsf{T} W_{\cdot,j,\ell} h_{j,t})$.

Starting from (4.11), we have

$$
\begin{aligned}
P(\mathbf{h}_{1:T}|\mathbf{x}_{1:T},\boldsymbol{\theta}) &= \frac{\prod_{t=1}^{T} f_t(\mathbf{x}_{1:T},\boldsymbol{\theta}) \prod_j g_t(\mathbf{x}_{1:T}, h_{j,t}, \boldsymbol{\theta})}{\prod_{t=1}^{T} f_t(\mathbf{x}_{1:T},\boldsymbol{\theta}) \sum_{\tilde{h}_{1,1}} \cdots \sum_{\tilde{h}_{N_h,1}} \cdots \sum_{\tilde{h}_{N_h,T}} \prod_{t=1}^{T} \prod_j g_t(\mathbf{x}_{1:T}, \tilde{h}_{j,t}, \boldsymbol{\theta})} \\
&= \frac{\prod_{t=1}^{T} \prod_j g_t(\mathbf{x}_{1:T}, h_{j,t}, \boldsymbol{\theta})}{\sum_{\tilde{h}_{1,1}} g_1(\mathbf{x}_{1:T}, \tilde{h}_{1,1}, \boldsymbol{\theta}) \cdots \sum_{\tilde{h}_{N_h,1}} g_1(\mathbf{x}_{1:T}, \tilde{h}_{N_h,1}, \boldsymbol{\theta}) \cdots \sum_{\tilde{h}_{N_h,T}} g_T(\mathbf{x}_{1:T}, \tilde{h}_{N_h,T}, \boldsymbol{\theta})} \\
&= \frac{\prod_{t=1}^{T} \prod_j g_t(\mathbf{x}_{1:T}, h_{j,t}, \boldsymbol{\theta})}{\prod_{t=1}^{T} \prod_j \sum_{\tilde{h}_{j,t}} g_t(\mathbf{x}_{1:T}, \tilde{h}_{j,t}, \boldsymbol{\theta})} = \frac{\prod_{t=1}^{T} \prod_j \exp(b_j h_{j,t} + \sum_{\ell=1}^{L} \mathbf{x}_{t+\ell-1}^{\mathsf{T}} W_{\cdot,j,\ell} h_{j,t})}{\prod_{t=1}^{T} \prod_j \sum_{\tilde{h}_{j,t}} \exp(b_j \tilde{h}_{j,t} + \sum_{\ell=1}^{L} \mathbf{x}_{t+\ell-1}^{\mathsf{T}} W_{\cdot,j,\ell} \tilde{h}_{j,t})} \\
&= \prod_{t=1}^{T} \prod_j P(h_{j,t}|\mathbf{x}_{1:T},\boldsymbol{\theta}) \ , \qquad (4.12)
\end{aligned}
$$

where we have used N_h to denote the number of hidden units.

Similarly, we can solve for the conditional distribution of the visible units given the hidden, to get

$$
P(\mathbf{x}_{1:T}|\mathbf{h}_{1:T},\boldsymbol{\theta}) = \prod_{t=1}^{T} \prod_i P(x_{i,t}|\mathbf{h}_{1:T},\boldsymbol{\theta}) \ . \qquad (4.13)
$$

Melody is encoded using a one-of-m representation, where at each time-step only the unit of the note being played is "on". The conditional distributions for the visible units are given by the softmax function

$$
P(x_{i,t}=1|\mathbf{h}_{1:T}) = \frac{\exp\left(c_i + \sum_{\ell=1}^{L} W_{i,\cdot,\ell} \mathbf{h}_{t-\ell+1}\right)}{\sum_q \exp\left(c_q + \sum_{\ell=1}^{L} W_{q,\cdot,\ell} \mathbf{h}_{t-\ell+1}\right)}, \qquad (4.14)
$$

For the hidden layer, we use binary logistic units, so that their conditional distributions are given by sigmoid functions

$$
P(h_{j,t}=1|\mathbf{x}_{1:T}) = \frac{1}{1+\exp\left(-b_j - \sum_{\ell=1}^{L} \mathbf{x}_{t+\ell-1}^{\mathsf{T}} W_{\cdot,j,\ell}\right)} \ . \qquad (4.15)
$$

4.2.1 Inference and Learning

Inference in the TC-RBM is easy, as the conditional distribution for the hidden state, $\mathbf{h}_{1:T}$, given an observed sequence, $\mathbf{x}_{1:T}$, factorises over time-steps and units. The computation of (4.14) and (4.15) can be performed efficiently by convolving along the time dimension the appropriate (shifted) slice of the weight array with the hidden and visible sequence respectively. The exact computations are detailed below. Note that we implement convolution using fast Fourier transforms — functions `fft2` and `ifft2` in MATLAB— which leads to computational savings compared to performing the additions and multiplications present in (4.14) and (4.15).

Let $\mathbf{r}_j$ be a $(L-1+T)$-dimensional column vector with the inputs hidden unit j receives across time, i.e. a vector with the input to the sigmoid function at each time-step (excluding the bias b_j for simplicity). The $L-1$ additional elements are due to boundary effects at the edges of the sequence (from Figure 4.2 we can see that the hidden sequence is $(L-1)$ time-steps longer than the visible sequence). Let N_x denote the number of visible units at each time-step, i.e. the size of the vocabulary, and let $\overleftrightarrow{\mathbf{x}}$ be a $(2L-2+T) \times N_x$ matrix, that contains an observed sequence $\mathbf{x}_{1:T}$ in the middle — each row is the one-of-N_x vector for a single time-step — and both edges of the sequence are padded with $L-1$ rows containing the $\mathbf{0}_{N_x}$ row vector. Finally, let $\mathbf{W}^{\mathsf{T}}_{\leftarrow,j,\leftarrow}$ be the $L \times N_x$ slice of the weight array $\mathbf{W}$ corresponding to hidden unit j, where we have reversed (shifted) the order of both the rows and columns. Then we have

$$\mathbf{r}_j = \overleftrightarrow{\mathbf{x}} \star \mathbf{W}^{\mathsf{T}}_{\leftarrow,j,\leftarrow} \ , \tag{4.16}$$

where the $\star$ operator performs discrete convolution, and returns only the parts computed without further zero-padding of the edges.

Similarly for the inputs to visible unit i, $\mathbf{r}_i$, let $\overleftarrow{\mathbf{h}}$ be a $(L-1+T) \times N_h$ matrix with the hidden configuration resulting from activating the hidden units given the inputs in (4.16) and let $\mathbf{W}^{\mathsf{T}}_{i,\cdot,\leftarrow}$ be the $L \times N_h$ slice of the weight array $\mathbf{W}$, where we have reversed only the order of the columns; in this case the oldest hidden time-step should be multiplied with the L-th slice of the weight array. Then we have

$$\mathbf{r}_i = \overleftarrow{\mathbf{h}} \star \mathbf{W}^{\mathsf{T}}_{i,\cdot,\leftarrow} \ . \tag{4.17}$$

The resulting inputs vector, $\mathbf{r}_i$, is a T-dimensional column vector.

Learning in the TC-RBM can be performed in a similar manner to the RBM, since the conditional distributions in (4.12) and (4.13) factorise, and can be sampled from efficiently. For instance, to obtain an estimator of the log-likelihood gradient, we would have to perform block Gibbs sampling, where we alternate between sampling the hidden units given the visibles and sampling the visible units given the hidden. When equilibrium is reached the expectation with respect to the model in 4.7 can be computed.

As sampling to equilibrium can be computationally expensive, in this work we use the Contrastive Divergence (CD) learning rule (Hinton, 2002) to update model parameters. The number of Gibbs step, k, for the CD rule is set by cross-validation.

The updates for the weights and the biases are given by

$$\Delta w_{i,j,\ell} = \epsilon(\langle x_{i,t+\ell-1} h_{j,t} \rangle_0 - \langle x_{i,t+\ell-1} h_{j,t} \rangle_k) \ , \tag{4.18}$$

$$\Delta c_i = \epsilon(\langle x_i \rangle_0 - \langle x_i \rangle_k) \ , \tag{4.19}$$

$$\Delta b_j = \epsilon(\langle h_j \rangle_0 - \langle h_j \rangle_k) \ , \tag{4.20}$$

where ϵ is the learning rate and $\langle \cdot \rangle_k$ represents an expectation with respect to the distribution defined by initialising the MCMC chain at a data vector and running block Gibbs sampling for k full steps.

4.3 Model Evaluation

To train a TC-RBM network we need to make certain decisions regarding the details of the training algorithm, but also the exact network architecture. The former refers to the values of learning meta-parameters, such as the learning rate, the mini-batch size, the inclusion of a sparsity constraint and if so, the level of sparsity and the number of Gibbs steps used in CD. The latter refers to the number of hidden units and the size of the filter that we apply, that is the length of the subsequence that each unit receives input from.

In this Section we analyse how TC-RBMs with different architectures perform under different training conditions. We compare model performance using the average next-step prediction log-likelihood acquired using a 10-fold cross-validation

procedure. The predictive distribution of the next time-step under the TC-RBM is approximated through the sampling procedure detailed in Algorithm 3. The probability of the visible vector at time-step t, $\mathbf{x}_t$, given the previous $t-1$ visible vectors, $\mathbf{x}_{1:t-1}$ can be written as

$$P(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \boldsymbol{\theta}) = \sum_{\mathbf{h}_{1:t}} P(\mathbf{x}_t|\mathbf{h}_{1:t}, \boldsymbol{\theta}) P(\mathbf{h}_{1:t}|\mathbf{x}_{1:t-1}, \boldsymbol{\theta}) \ . \tag{4.21}$$

Let $\{\mathbf{h}^s_{1:t}|s = 1, \ldots, S\}$ be S samples drawn from $P(\mathbf{h}_{1:t}|\mathbf{x}_{1:t-1}, \boldsymbol{\theta})$. Then a simple Monte Carlo estimate of the predictive distribution conditioned on previous observations can be computed as

$$P(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \boldsymbol{\theta}) \approx \frac{1}{S} \sum_{s=1}^{S} P(\mathbf{x}_t|\mathbf{h}^s_{1:t}, \boldsymbol{\theta}) \ , \tag{4.22}$$

However, we cannot easily draw samples from $P(\mathbf{h}_{1:t}|\mathbf{x}_{1:t-1}, \boldsymbol{\theta})$ and we approximate (4.22) by constructing a Markov chain that allows us to draw samples from $P(\mathbf{h}_{1:t+\tau}, \mathbf{x}_{t:t+\tau}|\mathbf{x}_{1:t-1}, \boldsymbol{\theta})$. The previous visible vectors, $\mathbf{x}_{1:t-1}$, are clamped to the data and the current and future visible vectors, $\mathbf{x}_{t:t+\tau}$, are randomly initialised. At each step, s, of the Markov chain we sample

$$\mathbf{h}^s_{1:t+\tau} \sim P(\mathbf{h}_{1:t+\tau}|\mathbf{x}_{1:t-1}, \mathbf{x}^{s-1}_{t:t+\tau}, \boldsymbol{\theta}) \ ,$$

using the factorisation in Equation (4.12) and subsequently sample

$$\mathbf{x}^s_{t:t+\tau} \sim P(\mathbf{x}_{t:t+\tau}|\mathbf{h}^s_{1:t+\tau}, \boldsymbol{\theta}) \ ,$$

using the factorisation in Equation (4.13). The τ additional steps are used to deal with the edges of the convolution. More specifically, the hidden units at time-step t receive input from the visible units at time-steps $t : t + L - 1$. Therefore, if we only use visible time-steps up to time t, then the remaining $L - 1$ time-steps are implicitly set to 0, which is not a valid configuration with respect to our data. In the following experiments we use $\tau = 2L - 1$.

Unless reported otherwise, we approximate the predictive log-likelihood using 50 hidden samples. These are collected by running 10 Gibbs chains, discarding the first 5 Gibbs steps in each chain and collecting the conditional probability distributions from the following 5 Gibbs steps.

Algorithm 3 Compute predictive probabilities under the TC-RBM.

This procedure approximates the conditional probability distribution of the next time-step, $\mathbf{x}_t$, given an observed context, $\mathbf{x}_{1:t-1}$, under the TC-RBM.

If the context is shorter than the filter size, L, then it is padded with zero vectors in the beginning.

Input: $\mathbf{x}_{1:t-1}$, $\boldsymbol{\theta}$, #initGibbsSamples, #GibbsSamples: S, #GibbsChains:C, #FreeSteps:τ

Due to boundary effects in the TC-RBM, we use τ free visible time-steps during this procedure. These are updated together with the t-th time-step, but are not used in the approximation.

for each Gibbs chain c **in** $1, \ldots, C$ **do**
  Initialise $\mathbf{x}^0_{t:t+\tau}$ randomly.
  for burn-in sample s **in** $1, \ldots,$ #initGibbsSamples **do**
    Sample $\mathbf{h}^s_{1:t+\tau} \sim P(\mathbf{h}^s_{1:t+\tau} | \mathbf{x}^{s-1}_{1:t+\tau}, \boldsymbol{\theta})$ (Eqns.4.12 and 4.15).
    Sample $\mathbf{x}^s_{t:t+\tau} \sim P(\mathbf{x}^s_{t:t+\tau} | \mathbf{h}^s_{1:t+\tau}, \boldsymbol{\theta})$ (Eqns. 4.13 and 4.14).
  end for
  Set $\mathbf{x}^0_{t:t+\tau} = \mathbf{x}^s_{t:t+\tau}$
  for each sample s **in** $1, \ldots, S$ **do**
    Sample $\mathbf{h}^s_{1:t+\tau} \sim P(\mathbf{h}^s_{1:t+\tau} | \mathbf{x}^{s-1}_{1:t+\tau}, \boldsymbol{\theta})$ (Eqns.4.12 and 4.15).
    Sample $\mathbf{x}^s_{t:t+\tau} \sim P(\mathbf{x}^s_{t:t+\tau} | \mathbf{h}^s_{1:t+\tau}, \boldsymbol{\theta})$ (Eqns. 4.13 and 4.14).
    Collect $p_{s,c} = P(\mathbf{x}^s_t | \mathbf{h}^s_{1:t+\tau}, \boldsymbol{\theta})$.
  end for
end for
Approximate $P(\mathbf{x}_t | \mathbf{x}_{1:t-1}, \boldsymbol{\theta}) \cong \frac{1}{C}\frac{1}{S} \sum_c \sum_s p_{s,c}$

4.3.1 Implementation Details

Table 4.1 shows a list of model and training meta-parameters, together with the values that we examine. Here we give a more detailed description of how some of these are implemented.

We use *filter size* to describe L, the length of the subsequence that the hidden units at each time-step receive input from. A filter size of 8 corresponds to a musical bar in our representation; we have 4/4 meter and time is discretised in

eighth notes.

A *fixed* learning schedule means that the learning rate is constant throughout training. In the *adaptive* learning schedule the learning rate is decreased by 10% every 25 epochs.

Table 4.1: Model and learning meta-parameters and the different values we examine. A grid search over the product space of these values is performed.

Parameter	Values
Number of hidden units	10, 50, 100
Number of Gibbs steps for CD	1, 5, 10
Filter size	4, 8
Learning rate	0.3, 0.5, 0.8
Learning schedule	fixed, adaptive
Mini-batch size	10 sequences, 20 sequences
Sparsity target	no sparsity constraint, 0.05, 0.1, 0.2

Due to the over-complete hidden representation in convolutional RBM architectures, it has been suggested that encouraging sparsity in the hidden unit activations is important and can facilitate learning (Lee et al., 2009; Norouzi et al., 2009). In this work we implement sparsity as described in Lee et al. (2008), where the objective function maximises the log-likelihood of the data, while applying a penalty term if the expected hidden unit activation is higher than a predefined level, p. For a training set comprising N_d sequences $\left\{\mathbf{x}^{(1)}_{1:T_1}, \cdots, \mathbf{x}^{(N_d)}_{1:T_{N_d}}\right\}$, the objective function can be written as

$$\underset{w_{i,j,\ell},b_j,c_i}{\arg\min} -\sum_{d=1}^{N_d}\left(\log \sum_{\mathbf{h}^{(d)}_{1:T_d}} P(\mathbf{x}^{(d)}_{1:T_d}, \mathbf{h}^{(d)}_{1:T_d}|\boldsymbol{\theta})\right) + \lambda \sum_{j=1}^{N_h}\left(p - \frac{1}{\sum_{d=1}^{N_d} T_d}\sum_{d=1}^{N_d}\sum_{t=1}^{T_d} \mathrm{E}\left[h^{(d)}_{j,t}|\mathbf{x}^{(d)}_{1:T_d}\right]\right)^2 \quad (4.23)$$

where λ is the strength of the regularisation term, p is the desired level of hidden unit activity and the expectation, $\mathrm{E}[\cdot]$, is taken over the conditional distribution of the hidden unit given the data. As in Lee et al. (2008), we compute the gradient for the regularisation term for each mini-batch and update only the biases, b_j, with respect to this gradient.

We should also note that there are certain implementation choices, which were made during preliminary experiments and are not analysed here. The weights,

$w_{i,j,\ell}$, are initialised using a Gaussian distribution, $\mathrm{N}(0, 0.01)$. The biases for the hidden units, b_j, are initialised to 0 and the biases for the visible units, c_i, are initialised using the log of the empirical marginal distribution. We use a weight decay of 0.0002. We set the $\lambda = 1$ for the regularisation term of the objective function in (4.23).

4.3.2 Analysing Model and Training Meta-Parameters

In the first experiment, we examine 7 model and training meta-parameters, each with 2 to 4 possible values. These are listed in Table 4.1. We perform a grid search over the product space of possible values, giving rise to 864 different configurations. For each configuration, we perform 10-fold cross-validation, where each network is trained for 50 epochs, and evaluate performance using the average next-step prediction log-likelihood. We do not recommend the coarse grid-search evaluation performed here for future research. Random search (Bergstra and Bengio, 2012) is a more efficient alternative with respect to the exploration of the meta-parameter space and has no computational or technical overhead. Additionally, we refer the reader to Hinton (2010) for a practical guide that discusses how to set the different meta-parameters when training Restricted Boltzmann Machines.

Table 4.2 shows the five best and five worst performing network configurations. The reported mean is the next-step prediction log-likelihood computed over all time-steps from all 10 folds. The reported standard deviation is computed over the mean log-likelihood for each fold. It shows how performance varies across different folds and it allows us to evaluate whether a configuration consistently performs well or badly, irrespective of the data subset used.

The highest log-likelihoods are achieved by networks with either 10 or 50 hidden units and a filter size of 8 time-steps, which are trained using the CD-1 objective, mini-batches of 10 sequences, a high learning rate and no sparsity. If we look at the worst performing networks, we can see that the main difference is that networks have 100 hidden units and are mainly trained using mini-batches with 20 sequences. The values for the rest of the model and learning parameters are repeated in the worst performing networks, i.e. we have high learning rates, filter sizes of 8 time-steps and no sparsity. However, we can note that there are also some networks trained using the least sparsity, where the hidden activity is restricted

Table 4.2: Best and worst performing network configurations. The 864 configurations are ranked according to the average next-step prediction log-likelihood, computed after training each network for 50 epochs. The mean is computed over all time-steps from all 10 folds. The standard deviation is computed across the 10 folds (not across time-steps) and shows how the performance of a configuration varies across different datasets (for both training and testing).

Best Performing Configurations (Top-5)							
Hidden Units	Filter Size	CD steps	Batch Size	Learning Rate	Learning Schedule	Sparsity	Mean ± St.Dev.
10	8	1	10	0.8	fixed	no	-1.8768 ± 0.05
10	8	1	10	0.8	adaptive	no	-1.8772 ± 0.05
50	8	1	10	0.8	fixed	no	-1.8874 ± 0.05
50	8	1	10	0.8	adaptive	no	-1.8918 ± 0.05
10	8	1	10	0.5	fixed	no	-1.9016 ± 0.05
Worst Performing Configurations (Bottom-5)							
Hidden Units	Filter Size	CD steps	Batch Size	Learning Rate	Learning Schedule	Sparsity	Mean ± St.Dev.
100	8	10	20	0.8	fixed	0.2	-2.8551 ± 0.50
100	8	1	20	0.5	adaptive	no	-2.7523 ± 0.50
100	8	1	20	0.5	fixed	no	-2.7507 ± 0.39
100	4	1	20	0.5	fixed	no	-2.6303 ± 0.34
100	8	10	10	0.8	fixed	0.2	-2.5986 ± 0.29

to 0.2.

For the best performing networks the standard deviation across folds is very low, suggesting that these configurations achieve high log-likelihoods in all the folds. For the worst performing networks the standard deviation is higher, suggesting that these configurations perform very poorly in some folds and somewhat better in others. However, the standard deviation is still fairly low, suggesting that performance using these configurations is bad in all folds. Note that the mean log-likelihood across all configurations is -2.1311.

To further analyse the results of this experiment, we present plots of the different learning and model parameters in Figures 4.3–4.9. Each figure shows the average prediction log-likelihood for the different values of a meta-parameter, plotted against all possible configurations for the remaining meta-parameters. The different values of the examined meta-parameter are colour-coded. These plots allows us

to evaluate whether a certain value always results on better performance, or whether specific settings are problematic and should be avoided. We first analyse performance with respect to each meta-parameter and then present a summary of our findings.

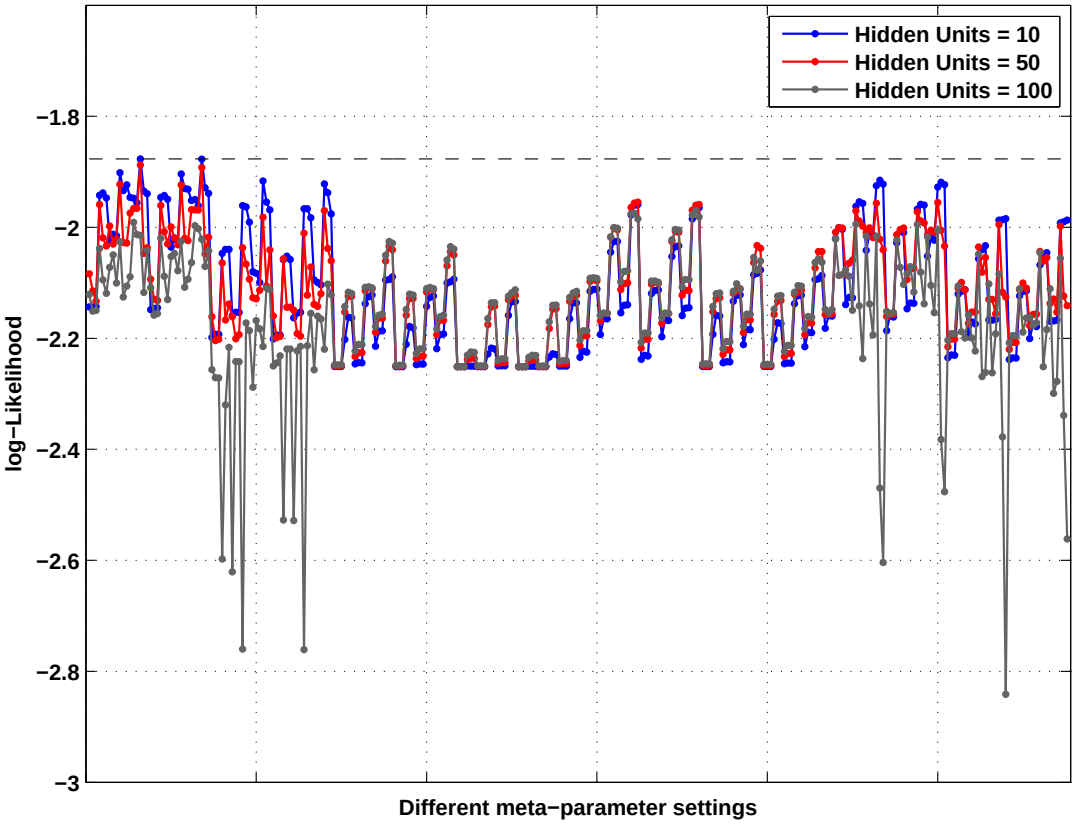


Figure 4.3: Average next-step prediction log-likelihood under TC-RBMs with 10, 50 and 100 hidden units, plotted against different settings for the 6 remaining meta-parameters. The networks are trained for 50 epochs. The average is computed over time-steps from all 10 folds.

4.3.2.1 Number of Hidden Units

Looking at Figure 4.3 we can see that the configurations with the highest log-likelihoods have either 10 or 50 hidden units (left part of the plot). On the other hand, we can immediately observe that there are roughly fifteen configurations, for which networks with 100 hidden units perform very poorly.

Overall we can distinguish two patterns of performance with respect to the number of hidden units. For a subset of configurations (left and right ends of the plot), we

have 10 hidden units giving the highest log-likelihood, with 50 hidden units giving slightly lower and 100 hidden units even lower and in some cases, considerably low. In the second subset of configurations (middle part of the plot), the log-likelihood achieved by networks with different numbers of hidden units is almost the same, with networks having 100 and 50 hidden units being marginally better in certain cases.

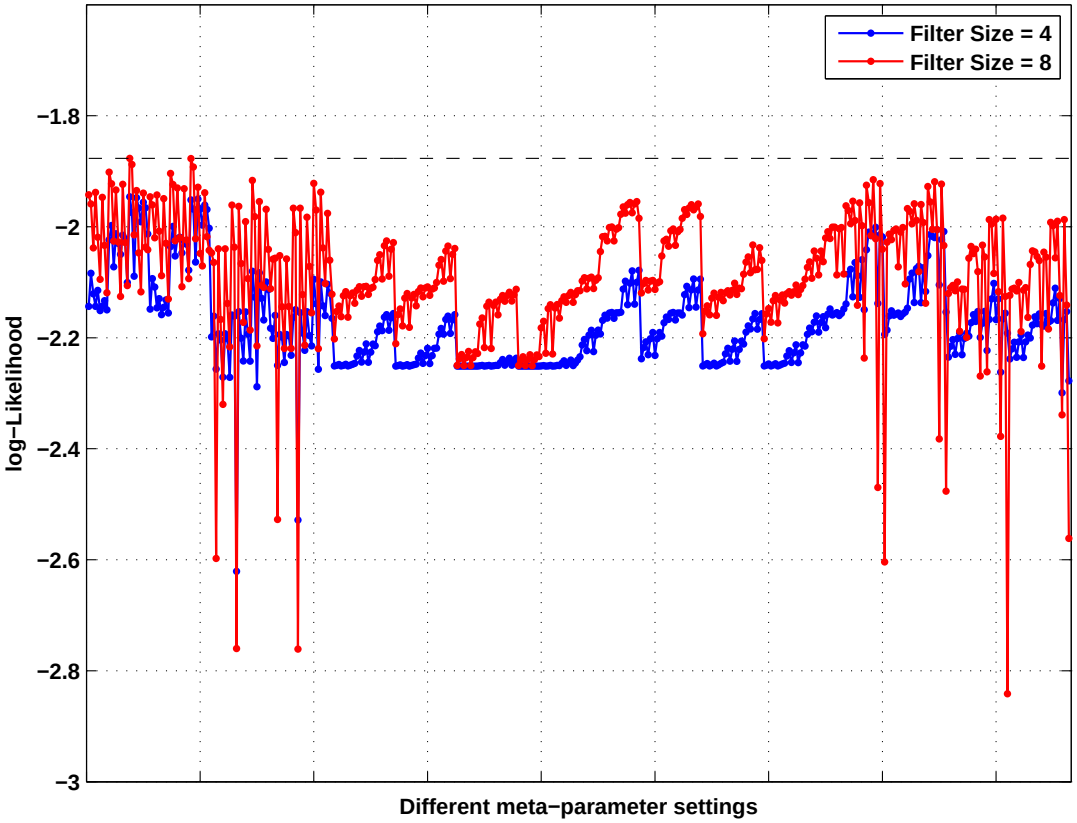


Figure 4.4: Average next-step prediction log-likelihood under TC-RBMs with filters "looking at" 4 and 8 visible time-steps, plotted against different settings for the 6 remaining meta-parameters. The networks are trained for 50 epochs. The average is computed over time-steps from all 10 folds.

4.3.2.2 Filter Size

Looking at Figure 4.4 we can see that for most of the configurations, networks with a filter size of 8 have higher log-likelihood than networks with a filter size of 4. The top performing configurations (left part of the plot) have filter size 8. However, looking at the fifteen worst performing configurations, we can see that

most of them occur with a filter size of 8. In some cases both filter sizes result in very poor performance, but there are certain configurations for which a filter size of 8 performs very poorly, whereas a filter size of 4 results in much higher log-likelihood. This suggests that for certain configurations using a large filter size can lead to bad local maxima.

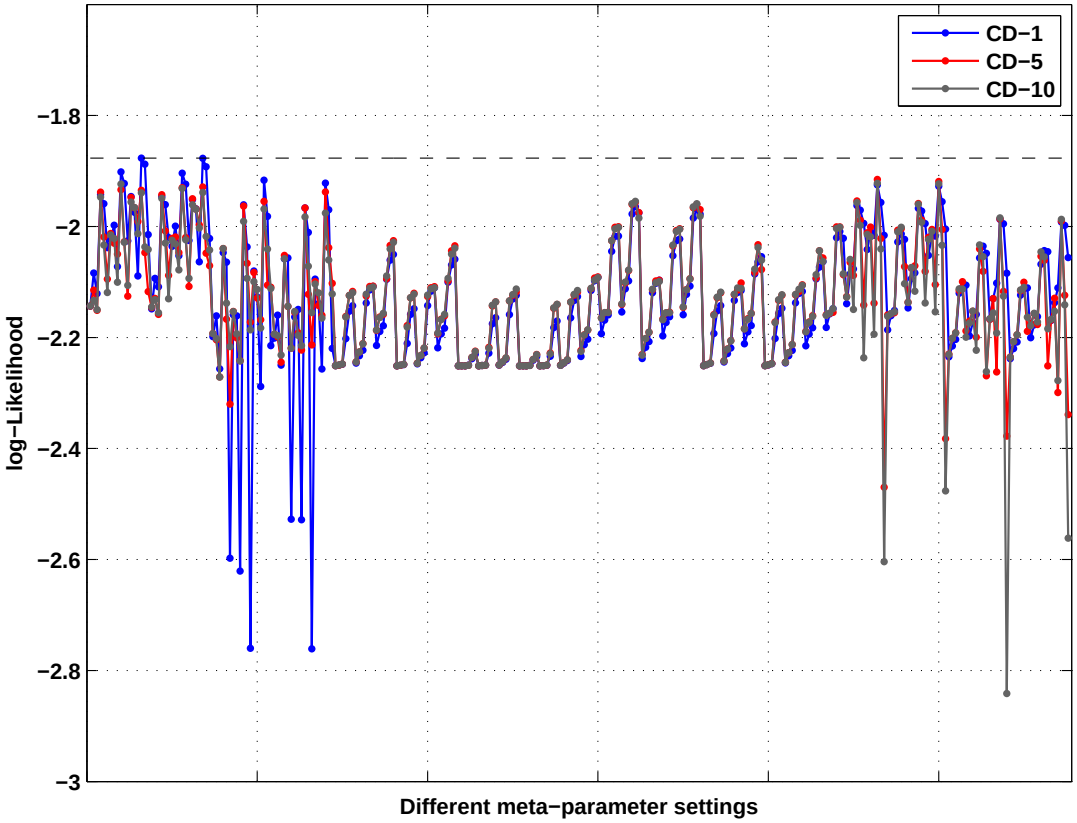


Figure 4.5: Average next-step prediction log-likelihood under TC-RBMs trained with CD-1, CD-5 and CD-10, plotted against different settings for the 6 remaining meta-parameters. The networks are trained for 50 epochs. The average is computed over time-steps from all 10 folds.

4.3.2.3 Gibbs steps in CD

There is no clear pattern between the performance of a network on the next-step prediction task and the number of Gibbs steps used for the Contrastive Divergence (CD) rule. More specifically, from Figure 4.5, we can see that in most cases, the three different settings for the number of CD steps achieve very similar log-likelihoods. The five top performing networks are trained using CD-1, however,

the equivalent networks trained with CD-5 and CD-10 are only marginally lower (left part of the plot).

Additionally, we can note that all three settings for CD steps appear in the worst fifteen configurations. In the left part of the plot, there are six configurations trained with CD-1, which have particularly low log-likelihood. In the right part of the plot we have four different configurations, which when trained with either CD-5 or CD-10, perform very poorly. However, note that using CD-5 in these four cases gives better results than using CD-10, suggesting that in problematic situations more steps lead to worse solutions. The overall pattern regarding the worst performing configurations suggests that using only one Gibbs step (we do not move away from the data) has somewhat different behaviour than using more steps.

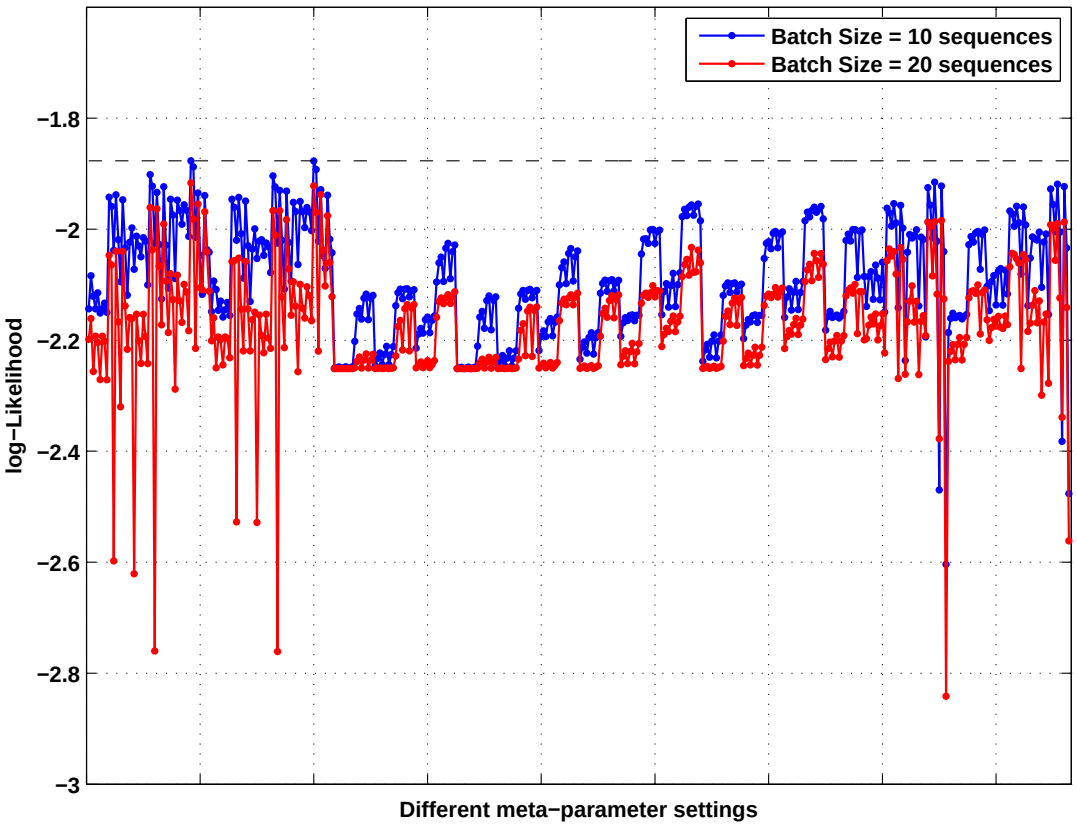


Figure 4.6: Average next-step prediction log-likelihood under TC-RBMs trained using mini-batches with 10 and 20 sequences, plotted against different settings for the 6 remaining meta-parameters. The networks are trained for 50 epochs. The average is computed over time-steps from all 10 folds.

4.3.2.4 Mini-batch Size

Using mini-batches of 10 sequences almost always results in higher log-likelihood than using mini-batches of 20 sequences. Additionally, from Figure 4.6 we can see that most of the worst performing configurations are trained with the larger mini-batch size, although for certain configurations the smaller mini-batch size also results in bad performance (right part of the plot).

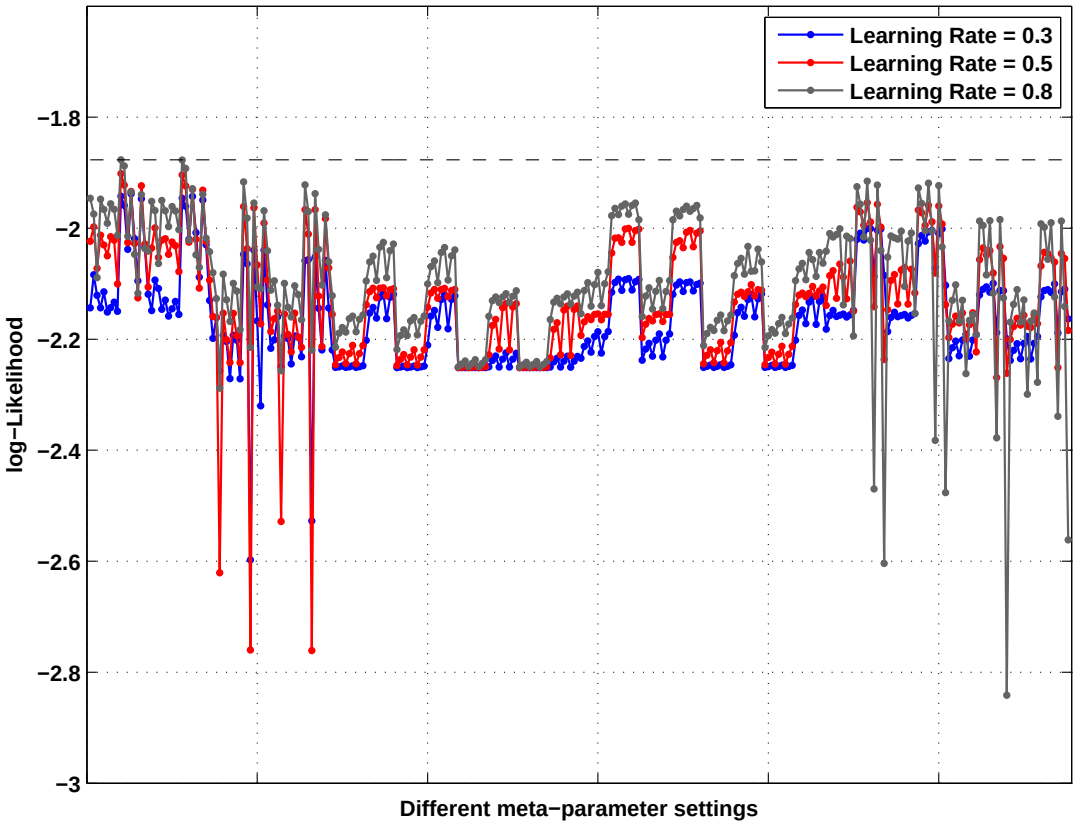


Figure 4.7: Average next-step prediction log-likelihood under TC-RBMs trained with learning rates 0.3, 0.5 and 0.8, plotted against different settings for the 6 remaining meta-parameters. The networks are trained for 50 epochs. The average is computed over time-steps from all 10 folds.

4.3.2.5 Learning Rate

Looking at Figure 4.7 we can see that for most configurations the higher the learning rate, the better the prediction performance. Regarding the worst performing configurations, we can see that all three settings can result in bad performance.

In these cases the best learning rate is subject to the configuration for the rest of the parameters.

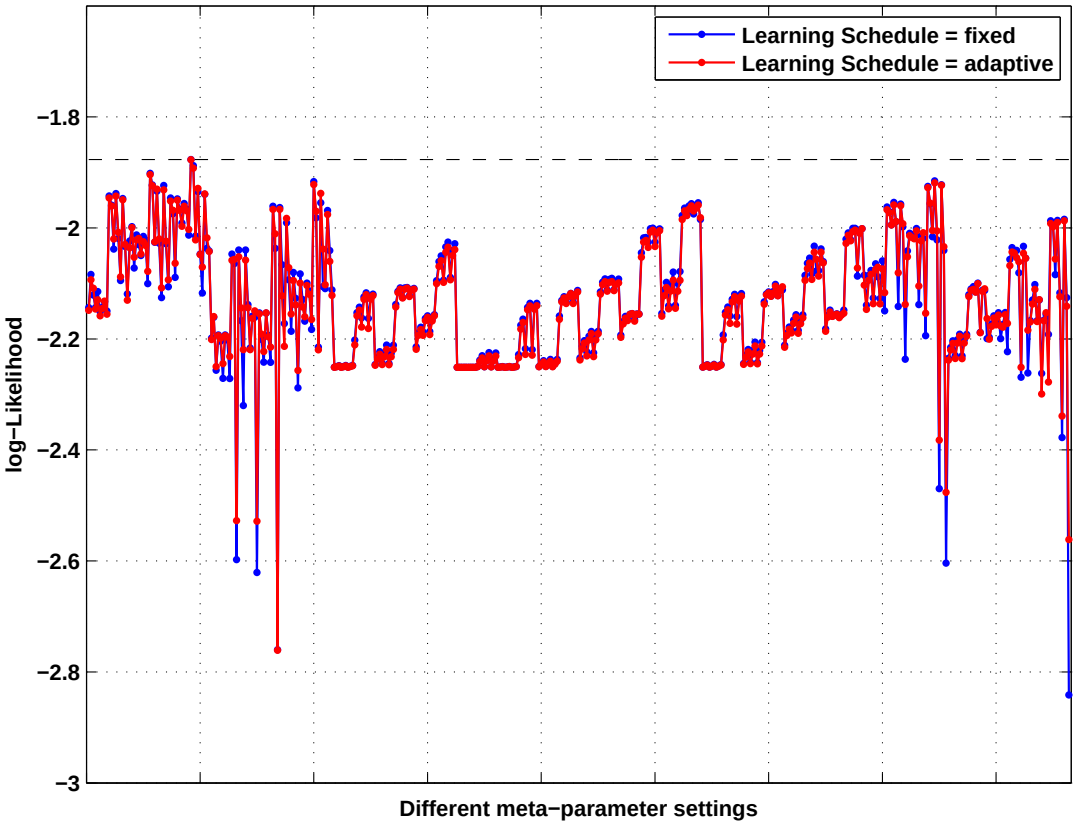


Figure 4.8: Average next-step prediction log-likelihood for TC-RBMs trained with "fixed" and "adaptive" learning rates, plotted against different settings for the 6 remaining meta-parameters. The networks are trained for 50 epochs. The average is computed over time-steps from all 10 folds.

4.3.2.6 Learning Schedule

Adapting the learning rate does not result in substantially different results. More specifically, looking at Figure 4.8 we can see that the two different schedules have almost the same performance for all the examined configurations. Regarding the worst performing networks, we can see that if the overall configuration is problematic, then both learning schedules result in bad performance; this is true in all cases. However, in the problematic cases, the adaptive learning schedule tends to give better performance than the fixed one, albeit poor comparing to better configurations for the rest of the parameters. We should note, that

adaptation of the learning rate occurs every 30 epochs and thus in this experiment, the learning rate has only been adapted once — we train the networks for 50 epochs. Therefore, the small performance differences between the two schedules are somewhat expected.

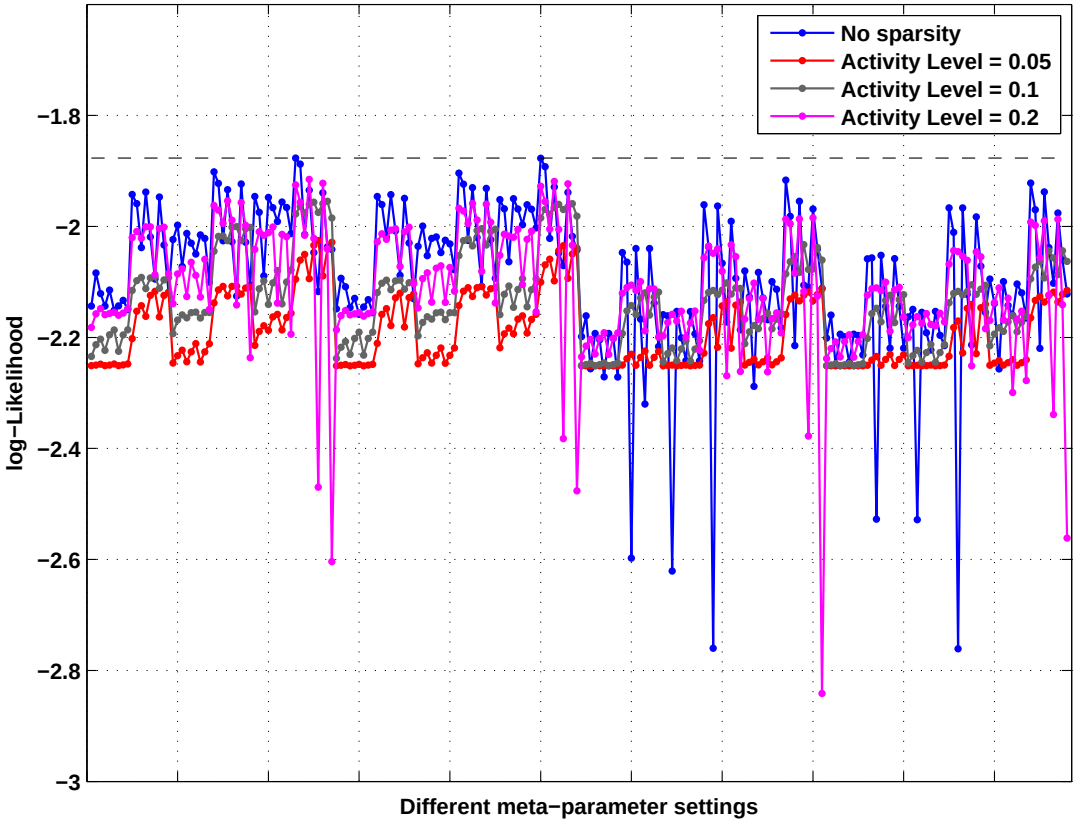


Figure 4.9: Average next-step prediction log-likelihood for TC-RBMs with different sparsity constraints, plotted against different settings for the 6 remaining meta-parameters. The networks are trained for 50 epochs. The average is computed over time-steps from all 10 folds.

4.3.2.7 Sparsity

From Figure 4.9 we can see that for most of the configurations imposing a sparsity constraint leads to lower log-likelihood and performance decreases as we use lower thresholds for the allowed activity in the hidden layer. However, we can note that the fifteen worst performing configurations correspond to networks with either no sparsity constraint or hidden activity set to 0.2, which suggests that restricting hidden unit activity to a low enough level can prevent training from visiting very

bad solutions.

4.3.2.8 Analysing model resource allocation

In the Time-Convolutional Restricted Boltzmann Machine the number of model parameters depends both on the number of hidden units and on the filter size. Additionally, networks trained with sparsity constraints are not "allowed" to use their full representational power, making the number of effective parameters, smaller than that of an unconstrained network. In this Section we compare configurations that have equivalent number of effective parameters. Our goal is to evaluate how resources should be allocated, if we want to maximise performance.

Hidden Units and Filter Size

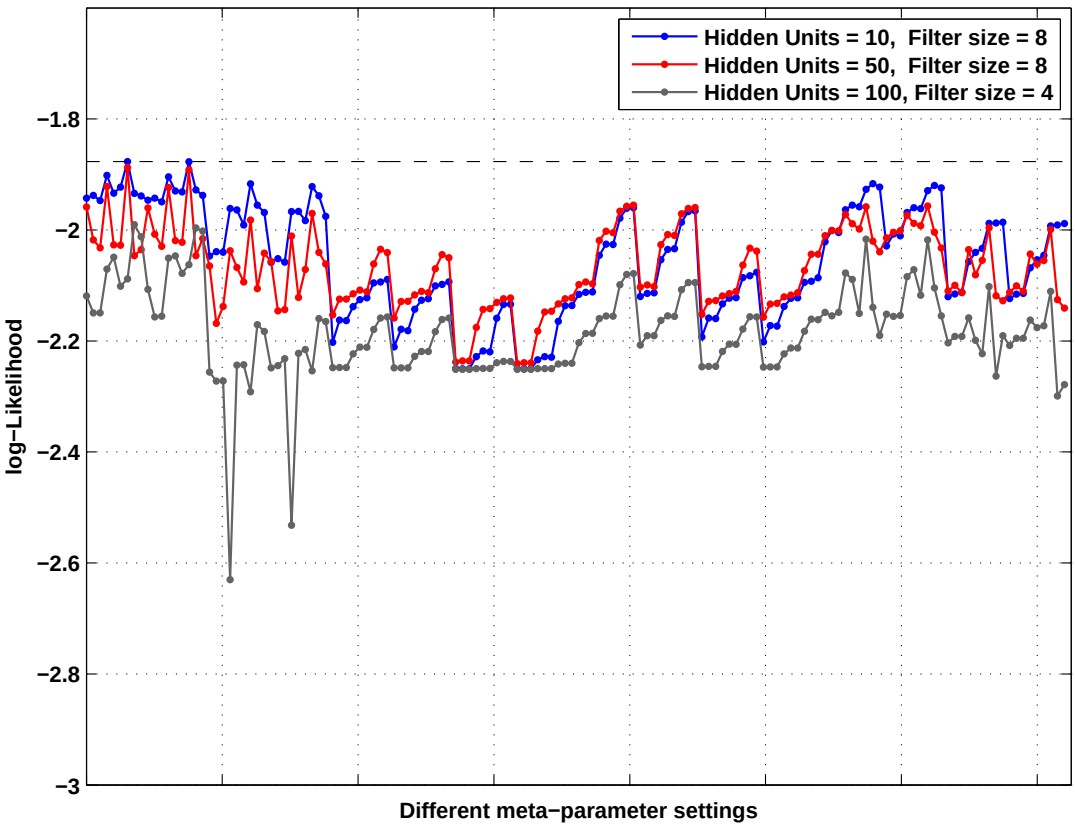


Figure 4.10: Average next-step prediction log-likelihood for TC-RBMs with the same numbers of weight parameters (red and gray), plotted against different settings for the 5 remaining meta-parameters. The networks are trained for 50 epochs. The average is computed over time-steps from all 10 folds.

In the TC-RBM, the weight array $\mathbf{W}$ has $(N_x \times N_h \times L)$ parameters, where N_x and N_h is the number of visible and hidden units, respectively, and L is the size of the filter. Additionally, we have $N_x + N_h$ parameters for the unit biases. In Figure 4.10 we present the performance of two TC-RBM configurations that have the same number of weight parameters. The gray line corresponds to networks with 100 hidden units and a filter size of 4, whereas the red line corresponds to networks with 50 hidden units and a filter size of 8. We additionally present results from networks with fewer parameters overall, but with a filter size of 8 (blue line). The plot shows that irrespective of what configuration we choose for the rest of the parameters, the networks with filter size 8 outperform networks with filter size 4, even when we have the same number of parameters by increasing the hidden units of the latter case. Note that networks with 100 hidden units and filter size 4 (gray line) have roughly 5 times more parameters than networks with 10 hidden units and filter size 8.

Hidden Units and Sparsity Level

In Figure 4.11 we present the performance of three TC-RBM configurations with an equivalent number of hidden units being allowed to turn "on" at every time-step. The blue line corresponds to a network with 10 hidden units and no sparsity constraint. The red line corresponds to networks with 50 hidden units and sparsity level set to 0.2, which means that on average we would expect to get 10 hidden units being "on" at each time-step. Finally, the gray line corresponds to networks with 100 hidden units and sparsity level set to 0.1, which again means that on average we get 10 hidden units "on" per time-step. This plot shows that networks with fewer hidden units and no sparsity constraint (blue line) have the highest performance irrespective of the values we use for the rest of the meta-parameters.

4.3.2.9 Summary of Findings

We examined 7 learning and model parameters for the TC-RBM, by performing 10-fold cross-validation on the product space of possible parameter values and comparing the different network configurations using the average prediction log-likelihood across folds. Regarding the model meta-parameters, using a higher number of hidden units is not particularly beneficial; in many cases networks with only 10 hidden units achieve the highest log-likelihood and there are roughly fifteen

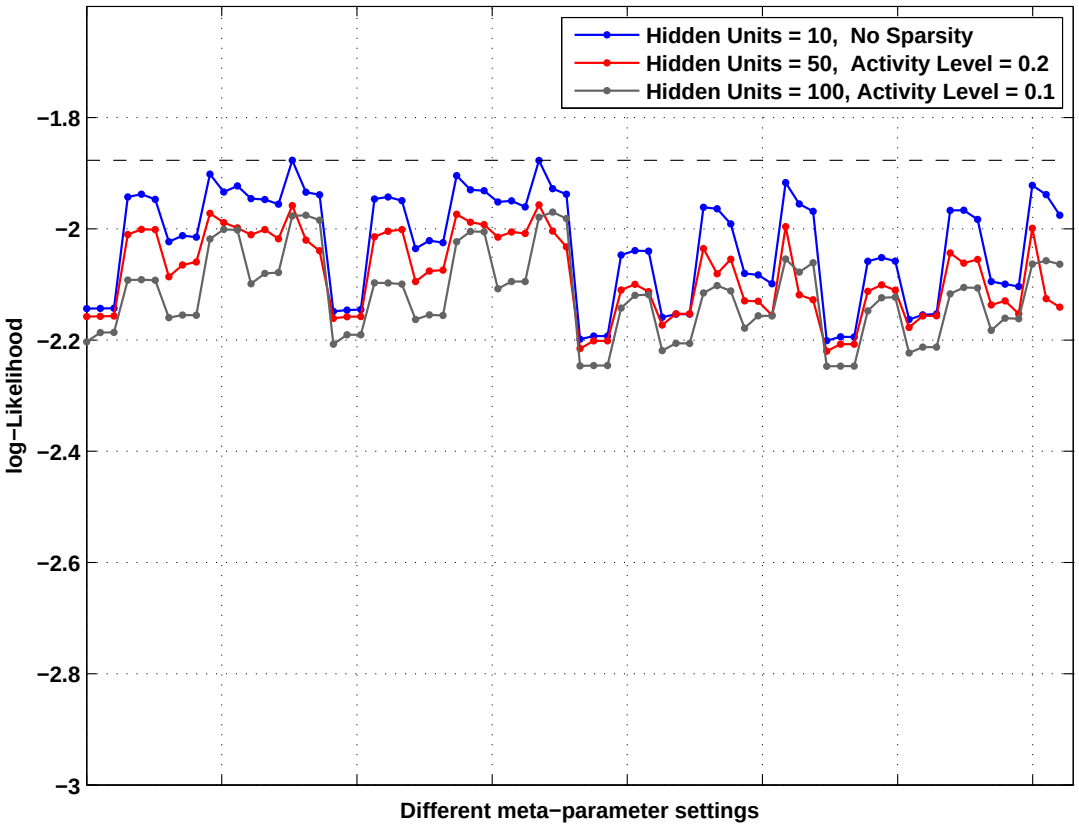


Figure 4.11: Average next-step prediction log-likelihood for TC-RBMs with an equivalent number of hidden units being allowed to turn "on" at every time-step. These are plotted against different settings for the 5 remaining meta-parameters. The networks are trained for 50 epochs. The average is computed over time-steps from all 10 folds.

configurations for which networks with 100 hidden units perform particularly badly. On the other hand, increasing the number of model parameters by using a larger filter size is advantageous, with the larger filter size (8 Vs 4 time-steps) giving consistently better results, with the exception of the fifteen worst performing configurations, which correspond to the larger filter size.

Regarding the training meta-parameters, using the smallest mini-batch size (10 sequences), the highest learning rate (0.8) and no sparsity constraint, result in higher log-likelihood for most of the cases. For the number of CD steps and learning schedule we do not get an overall winner, with performance being very similar for the different settings of these parameters.

These observations should be interpreted in conjunction with the number of training epochs that we have used. Due to the large number of possible configurations

Table 4.3: Model and learning meta-parameters and the different values we examine with respect to the training epochs. A grid search over the product space of these values is performed.

Parameter	Values
Number of hidden units	10, 50, 100
Number of Gibbs steps for CD	1, 5
Filter size	8
Learning rate	0.5, 0.8
Learning schedule	adaptive
Mini-batch size	10 sequences
Sparsity target	no sparsity constraint, 0.1

and the associated computational cost, we have evaluated these networks after training them for only 50 epochs. We generally expect that training in networks with many parameters will take longer to converge and this could explain why the lowest performance is attained by networks with 100 hidden units, filter size of 8 and no sparsity constraint or hidden activity constrained to a fairly high value. Additionally, the lowest performance corresponds to networks with the larger mini-batch size (20 sequences), i.e. when the model parameters (weights and biases) are updated fewer times in each epoch. Finally, the small number of epochs may also explain why the highest learning rate is preferable. We examine performance with respect to the training epochs in the next Section.

4.3.2.10 Training Epochs

In this Section we examine network performance with respect to the number of training epochs. We select a subset of the previously examined configurations and train the networks for 500 epochs, evaluating performance every 50 epochs. Again performance is measured as the average next-step prediction log-likelihood on the 10-fold cross-validation procedure. The configurations we examine are presented in Table 4.3.

Figure 4.12 shows the average next-step prediction log-likelihood of the networks plotted against the number of training epochs. For most networks, performance increases sharply in the first 50 to 100 epochs and then levels off, showing only

marginal changes. Networks trained with CD-5 and no sparsity constraints (red solid lines) result in pathological situations. This is true irrespective of the number of hidden units and the learning rate used, although the problem becomes bigger with more hidden units and larger learning rate. The networks improve initially, but then move to very bad solutions from which they don't recover completely. The problem is also evident when we apply the sparsity constraint (red dashed lines), though in this case the decay appears at a later stage and is more gradual.

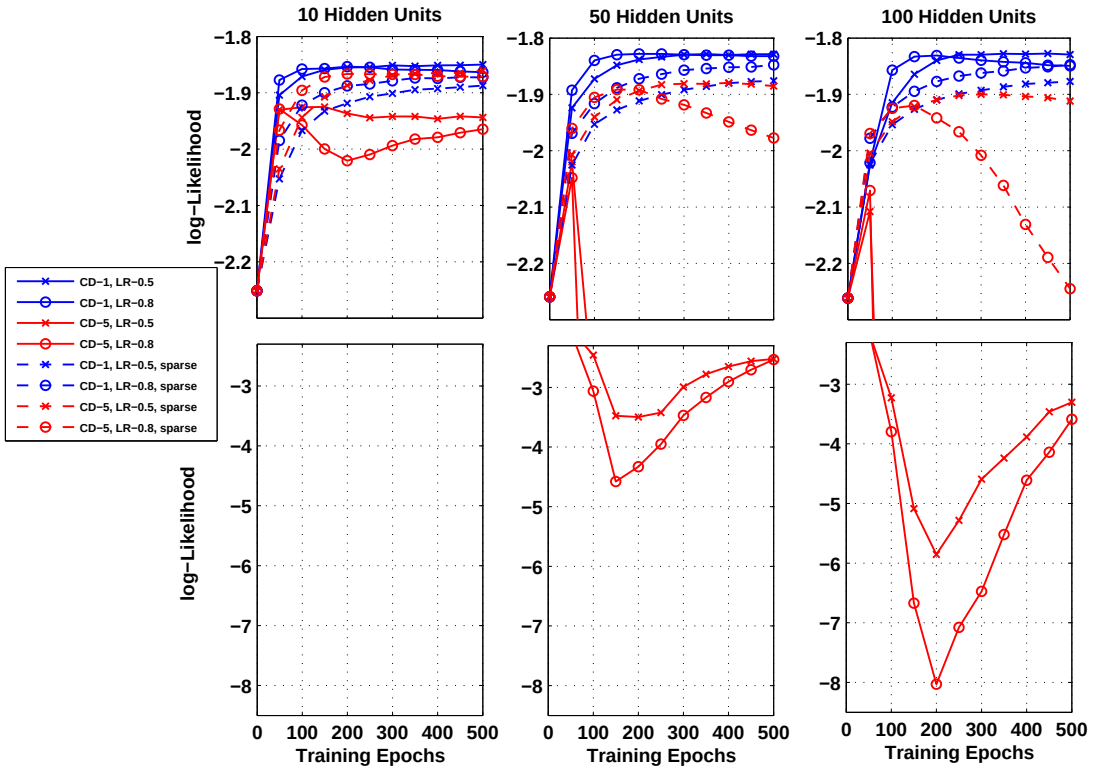


Figure 4.12: Average next-step prediction log-likelihood under TC-RBMs with different configurations, plotted against the number of training epochs. The average is computed over time-steps from all 10 folds. Note that to facilitate visualisation the scaling of the upper and lower part of the 3 plots is different.

Networks trained with CD-1 (blue lines) perform better overall and exhibit a more stable behaviour. In this case performance increases as we train the networks longer, although the increase gradually levels off. An exception to this are networks trained with a larger learning rate (0.8) and no sparsity constraints (blue solid lines with circles). For these networks performance starts to gradually decrease after a few hundred epochs.

Sparse networks (dashed lines) perform worse than their non-sparse counterparts,

excluding the pathological situations mentioned earlier. The larger learning rate gives much better performance in the beginning of training, but for the non-sparse networks, it leads to worse solutions. This combined with our observations above, suggests that more sophisticated learning rate adaptation schemes could be very beneficial.

Finally, if we compare the three plots, we can observe that performance in networks with more hidden units increases more slowly. In fact, when we look at performance after 50 epochs, networks with 10 hidden units are the best. However, networks with 50 units and even more so networks with 100 units show a big increase in performance when we move from 50 to 100 epochs, suggesting that using more hidden units is beneficial, provided that the networks are trained for longer.

4.3.3 Learning Musical Features

Besides the quantitative evaluation, we are also interested in assessing the ability of the TC-RBM to identify and represent the statistical regularities of the data. As discussed earlier, the TC-RBM learns a distributed representation of the input space; a set of latent features that are "on" or "off" depending on the input signal. Here we demonstrate that these features are music descriptors extracted from the data and convey information regarding music components such as scale, octaves and chords. For the purpose of this analysis, we use a sparse TC-RBM network with 50 hidden units and filter size of 8 time-steps, which is trained on music pieces from the key of G.

In the TC-RBM each hidden unit is connected with all the visible units from 8 consecutive time-steps. This gives rise to a 26×8-dimensional filter for each hidden unit, where the filter for hidden unit j is the slice $\mathbf{W}_{\cdot,j,\cdot}$ of the weight array.

The filters corresponding to 6 different hidden units from the learned TC-RBM are depicted in Figure 4.13. We can notice that all units prefer visible configurations with notes from the G major scale to be "on", but have various degrees of selectivity and respond to different subsets of these notes in different positions. Remember that the G-major key signature has one sharp, $F^{\#}$, so the scale notes are: $GABCDEF^{\#}G$.

For instance, filter 6 is fairly broad and may respond to several different con-

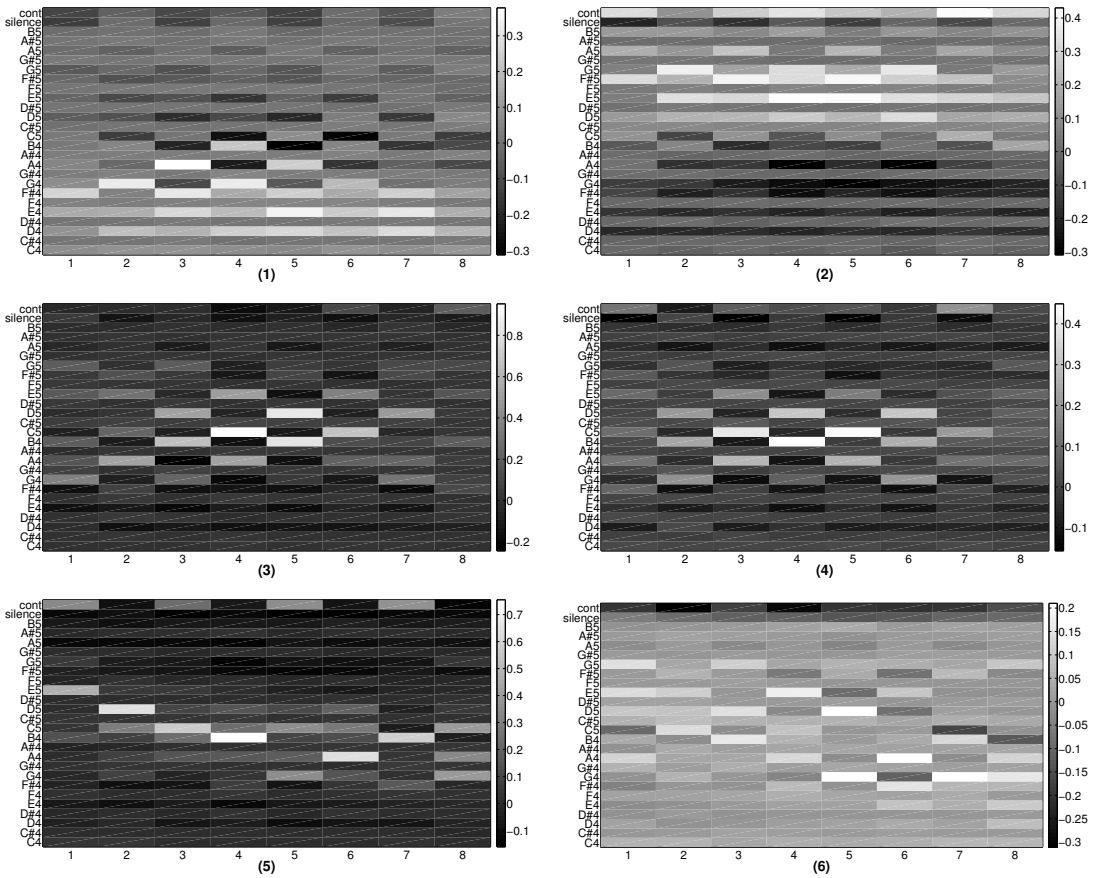


Figure 4.13: Weight filters for 6 different hidden units of a sparse TC-RBM network trained on pieces from the key of G. All units prefer notes from the G major scale to be "on". Filters 1 and 2 respond to similar patterns, but operate in the lower and higher octave respectively. Filters 3 and 4 respond to notes from either the $Gmaj$ or the Am triad in alternate time-steps. Filter 5 is highly selective to a specific motif, whereas filter 6 responds to several configurations of the scale notes.

figurations of notes from the G major scale, whereas filter 5 is highly selective, responding primarily to the downwards-upwards movement $EDCBGAB$ through the scale and certain variations of it.

An interesting property of the top two filters is their relation with respect to the octave. Both units respond to similar music phrases. For instance, both units respond to the motif $F^{\#}GAB$ starting at either position 1 or 3. However, the left unit operates in the lower octave ($C4$-$B4$), whereas the right one operates in the higher octave ($C5$-$B5$).

Another interesting property is the relation of the filters to chords that are

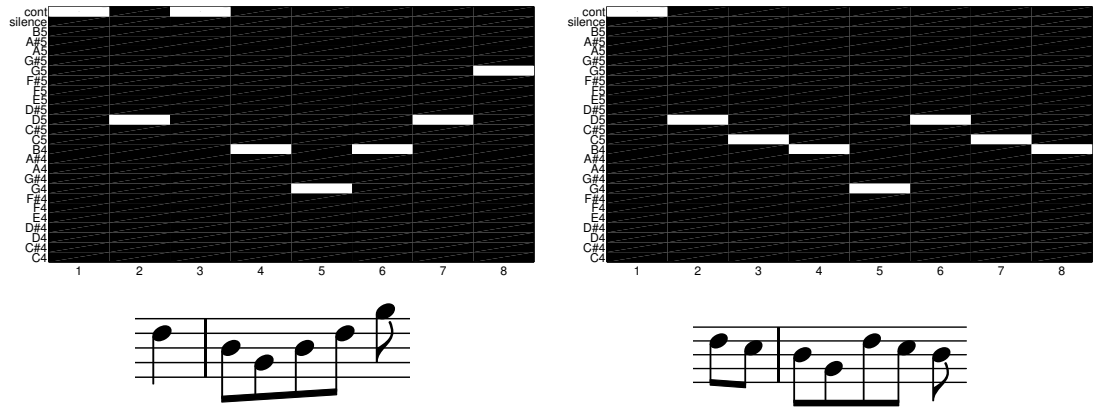


Figure 4.14: Two different visible configurations that frequently turn 'on' the hidden unit corresponding to filter 5 from Figure 4.13 during sampling. The visible configurations are also depicted in the musical score. Each configuration contains a different variation of the motif $D{*}BG$ in positions 2 to 5, which is prominent in filter 5.

commonly found in melodies in the key of G-major. In several filters, the preferred subset of notes at each time-step corresponds to alternate scale notes of G-major, which would be used to construct three-note chords. This property is particularly prominent in filters 3 and 4. For instance in filter 3, the preferred subset of notes at odd time-steps corresponds to the notes of the $Gmaj$ triad (GBD), whereas at even time-steps to the notes of the Am triad (ACE).

In order to better understand how the filters behave, we also look at visible configurations that tend to activate a hidden unit during sampling. Figure 4.14 shows two such visible configurations for the hidden unit corresponding to filter 5. Although the two configurations seem fairly different, they both contain the motif $D{*}BG$ in positions 2 to 5 with either a pass through C or 'continuation' of D in position 3. Filter 5 is highly responsive to this motif, and although time-steps 6 to 8 in the visible configurations are not highly preferable, the unit is still very likely to turn "on".

Overall, we can see that the learned filters encode familiar musical movements, such as arpeggios and scales, where these are loosely defined as groups of subsequent scale notes, either going up or going down the scale. However, the interesting and potentially powerful characteristic of the TC-RBM representation is that it also encodes and groups together many possible transformations of these movements. Therefore, in contrast to the kind of representation learned by the dictionary-based

predictors in Chapter 3, the motifs learned by the TC-RBM are not fixed; the TC-RBM filters group together musically sound variations of motifs, thus encoding possible note substitutions, merging and splitting. This is very advantageous when modelling music, given its highly complex and ingenious nature, and also allows for more genuine music generations.

4.4 Comparative Analysis

In this Section we compare model performance using the next-step prediction log-likelihood of the held-out testing set and the Maximum Mean Discrepancy (MMD) between testing pieces and model generations, computed using the mismatch string kernel. From the experiments described in the previous Section we selected the best performing networks with 10, 50 and 100 hidden units with no sparsity constraint and with sparsity level set to 0.1. The exact configurations of these 6 networks are shown in Table 4.4.

Table 4.4: The optimal meta-parameter settings for the 6 TC-RBM networks used for model comparison.

	Optimal Setting
TCRBM-10	$N_h = 10$, $L = 8$, CD-1, $\epsilon = 0.5$ adaptive, batch $= 10$, epoch $= 500$
TCRBM-50	$N_h = 50$, $L = 8$, CD-1, $\epsilon = 0.8$ adaptive, batch $= 10$, epoch $= 250$
TCRBM-100	$N_h = 100$, $L = 8$, CD-1, $\epsilon = 0.5$ adaptive, batch $= 10$, epoch $= 450$
TCRBM-10.sp	$N_h = 10$, $L = 8$, CD-5,$\epsilon = 0.5$ adaptive, batch $= 10$, epoch $= 500$, $p = 0.1$
TCRBM-50.sp	$N_h = 50$, $L = 8$, CD-1,$\epsilon = 0.8$ adaptive, batch $= 10$, epoch $= 500$, $p = 0.1$
TCRBM-100.sp	$N_h = 100$, $L = 8$, CD-1,$\epsilon = 0.8$ adaptive, batch $= 10$, epoch $= 500$, $p = 0.1$

We report results that are averaged across different folds. More specifically, for each network configuration we use the 10 trained models from the cross-validation experiment above to compute the evaluation metrics on the testing data and we report the mean and the standard deviation of the results across folds.

4.4.1 Next-Step Prediction

Table 4.5 shows the next-step prediction log-likelihood for the best performing TC-RBM networks. For the purpose of comparison we have included the results using

the empirical marginal distribution and the models examined in Chapter 3. As already mentioned, in the TC-RBM, the predictive probabilities are approximated using (4.22). Here we approximate these probabilities using 50, 150 and 250 hidden configuration samples, collected by running 10, 30 and 50 Gibbs chains, respectively, discarding the first 5 steps in each chain and using the next 5 Gibbs steps for the approximation.

Table 4.5: Average next step prediction log-likelihood of the testing data under different models. For each model type, we use the 10 trained models from cross-validation and compute the average next-step prediction log-likelihood of the testing data (64 pieces) under each of these models. The mean and standard deviation are computed across the results from the 10 trained models. Bold: Best TC-RBM configuration. TC-RBM networks with more hidden units and no sparsity perform better, but not as good as the variable-length models. Log-likelihood increases as we increase the number of samples used to approximate (4.22). See text for more details.

	Mean ± St.Dev.		
Empirical Marginal	-2.2217 ± 0.0009		
VMM	-1.7437 ± 0.0029		
Dirichlet-Bigram	-1.9171 ± 0.0019		
Dirichlet-VMM.Sh	-1.6084 ± 0.0030		
Dirichlet-VMM.De	-1.5663 ± 0.0040		
	50 Samples	150 Samples	250 Samples
TCRBM-10	-1.8127 ± 0.0058	-1.7965 ± 0.0056	-1.7931 ± 0.0056
TCRBM-50	-1.7873 ± 0.0065	-1.7681 ± 0.0061	-1.7641 ± 0.0060
TCRBM-100	-1.7949 ± 0.0038	-1.7689 ± 0.0035	**-1.7635** ± 0.0040
TCRBM-10.sp	-1.8241 ± 0.0450	-1.7982 ± 0.0441	-1.7929 ± 0.0444
TCRBM-50.sp	-1.8028 ± 0.0372	-1.7735 ± 0.0383	-1.7671 ± 0.0386
TCRBM-100.sp	-1.8096 ± 0.0382	-1.7751 ± 0.0372	-1.7683 ± 0.0368

The first thing we can observe is that in all the TC-RBM networks performance increases as we increase the number of samples used in Equation (4.22) to approximate the conditional probability distribution of the next time-step given the previous observations. Additionally, the improvement is higher for networks with more hidden units, which is expected as the number of possible hidden configurations is exponentially large in the number of hidden units and therefore for a given number of samples the approximation is poorer for networks with many hidden units. Furthermore, the improvement is substantial as we move from 50 to

150 samples, but becomes much smaller when we move from 150 to 250 samples.

The highest next-step prediction log-likelihood is achieved by the TC-RBM with 100 hidden units and no sparsity constraint. On the whole, networks with more hidden units perform better and networks with no sparsity constraint have higher log-likelihood than their sparse counterparts.

Comparing to the dictionary-based predictors from Chapter 3, the TC-RBMs perform better than the Dirichlet-Bigram, but are worse than any of the variable-length models. One advantage of the variable-length Markov models is their ability to use shorter or longer contexts depending on whether these provide useful information or not. On the other hand, the TC-RBM is modelling subsequences of fixed length, although information from different parts of the sequence is propagated through the hidden units.

Another limitation of the TC-RBM is that it is not optimised for prediction. Our approximation of the conditional probability distribution of the next step given the past under the TC-RBM (Algorithm 3), involves a block Gibbs sampling procedure, where the beginning of a sequence is clamped to the observed context and the rest of the sequence is initialised randomly and sampled as a whole at each Gibbs iteration. Therefore, due to the convolutional structure of the model, the time-step we are trying to predict receives information not only from the past, but also from the future which can drive the samples into different energy basins. These energy basins can correspond to "compatible" continuations of a music phrase, but do not necessarily match the true continuation.

4.4.2 Maximum Mean Discrepancy of String Kernels

In this Section we evaluate model performance by comparing how similar model samples are to testing sequences. As in Chapter 3, we generate 10 sets of samples from each TC-RBM configuration, where each set is sampled from a network trained on a different data fold. We then compute the Maximum Mean Discrepancy between each set of samples and the set of testing sequences, using the $(4, 1)$ mismatch kernel and the normalised $(4, 1)$ mismatch kernel and report the mean and the standard deviation of the squared MMD across the 10 sets of samples. As before, each set comprises 64 generated sequences having the same length as the 64 testing sequences and we use only the first half of each testing sequence

(and each sample) to compute the string kernels, in order to avoid results being dominated by exact repetitions. Additionally, we compute the MMD between each set of samples and 64 training pieces randomly selected from the data used to train each model and examine if the MMD is substantially lower in this case.

Table 4.6: Mean, standard deviation, minimum and maximum of the estimated squared MMD between testing sequences and samples generated from different models. Results for the $(4, 1)$ and the normalised $(4, 1)$ mismatch kernels. In the non-sparse TC-RBMs performance increases as we increase the number of hidden units, but is never as good as that of the Dirichlet-VMMs. In the sparse TC-RBMs performance varies a lot across folds indicating that some networks are stuck in bad local optima. See text for more details.

	$(4, 1)$ Mismatch Kernel			Normalised $(4, 1)$ Mismatch Kernel		
	Mean $\pm$ St.Dev.	Min	Max	Mean $\pm$ St.Dev.	Min	Max
Uniform $\mathcal{U}(\lvert\Sigma\rvert)$	411.69 ± 1.31	409.34	413.13	0.1657 ± 0.0011	0.1639	0.1668
EmpMarg	129.89 ± 10.13	116.04	147.08	0.0644 ± 0.0054	0.0543	0.0723
Training Data	6.22 ± 4.58	1.33	16.71	0.0030 ± 0.0015	0.0002	0.0048
VMM	173.58 ± 8.56	159.81	187.84	0.0269 ± 0.0046	0.0200	0.0360
Dir-Bigram	91.81 ± 6.10	83.67	102.18	0.0373 ± 0.0024	0.0342	0.0407
Dir-VMM.Sh	9.20 ± 4.97	2.71	19.79	0.0257 ± 0.0046	0.0163	0.0342
Dir-VMM.De	10.04 ± 5.23	2.18	18.71	0.0256 ± 0.0034	0.0205	0.0296
TCRBM-10	74.75 ± 13.49	60.63	101.61	0.0774 ± 0.0094	0.0639	0.0914
TCRBM-50	54.14 ± 6.12	44.65	66.24	0.0492 ± 0.0060	0.0415	0.0605
TCRBM-100	48.45 ± 3.81	39.30	52.19	0.0369 ± 0.0042	0.0299	0.0423
TCRBM-10.sp	188.51 ± 175.81	57.38	640.17	0.1278 ± 0.1204	0.0419	0.4393
TCRBM-50.sp	159.43 ± 94.36	43.82	354.32	0.1043 ± 0.0574	0.0459	0.2316
TCRBM-100.sp	158.25 ± 92.25	40.08	307.62	0.0994 ± 0.0471	0.0455	0.1959

Table 4.6 shows the mean and standard deviation of the estimated squared MMD between the testing sequences and the 10 sets of model samples using two different kernels. In order to compare the performance of the TC-RBM with other models we additionally report results using: a) a Uniform distribution – no information regarding the structure of the data, b) the empirical marginal distribution of the training data – no information regarding the temporal structure, c) the Dirichlet-Bigram – models first-order Markov dependencies, d) the VMM and Dirichlet-VMM models examined in Chapter 3 – all model variable-order Markov dependencies and e) the training data – drawn from the same true theoretical probability distribution as the testing data and act as an indicator of the best

Table 4.7: Mean, standard deviation, minimum and maximum of the estimated squared MMD between training sequences and samples generated from different models. Results for the $(4, 1)$ and the normalised $(4, 1)$ mismatch kernels.

	$(4,1)$ Mismatch Kernel			Normalised $(4,1)$ Mismatch Kernel		
	Mean ± St.Dev.	Min	Max	Mean ± St.Dev.	Min	Max
TCRBM-10	101.50 ± 12.95	78.30	123.11	0.1015 ± 0.0127	0.0849	0.1188
TCRBM-50	65.91 ± 18.59	46.57	105.13	0.0695 ± 0.0132	0.0514	0.0919
TCRBM-100	54.09 ± 15.90	39.96	88.46	0.0539 ± 0.0068	0.0430	0.0628
TCRBM-10.sp	186.65 ± 180.99	46.82	626.21	0.1406 ± 0.1187	0.0337	0.4381
TCRBM-50.sp	176.23 ± 113.93	42.55	358.60	0.1232 ± 0.0611	0.0343	0.2405
TCRBM-100.sp	181.68 ± 120.45	31.63	382.12	0.1201 ± 0.0526	0.0440	0.2100

performance that can be achieved given that we use a finite amount of sequences.

Looking at Table 4.6 we can see that all TC-RBM networks outperform the Uniform distribution, but are inferior to the Dirichlet-VMM models. Comparing Table 4.6 with Table 4.7, which contains the results of the MMD evaluation using training sequences and model samples, we can see that the two are equivalent, with the latter being slightly higher in most cases.

To facilitate visualisation of the comparative performance we provide barplots of the squared MMD for different models in Figures 4.15 and 4.16 for the $(4, 1)$ mismatch kernel and the normalised $(4, 1)$ mismatch kernel, respectively. In these barplots we exclude the squared MMD for the Uniform distribution.

Looking at Figure 4.15, we can see that the TC-RBM networks without sparsity constraints outperform the empirical marginal distribution, the Dirichlet-Bigram and the VMM, but are worse than the Dirichlet-VMM models. Performance increases as we increase the number of hidden units, although the improvement from 50 to 100 hidden units is smaller than the improvement from 10 to 50. Additionally, the performance of the non-sparse networks is fairly robust across folds, with the squared MMD having low standard deviation, especially when we use more than 10 hidden units.

On the other hand, we can see that the standard deviation is very high in the case of sparse TC-RBM networks. Performance is very unstable across folds, with the minimum value attained being comparable, and typically smaller, to the one from the equivalent non-sparse networks, but with the maximum value being close or

even worse than using samples from a Uniform distribution (Table 4.6), which indicates that sparse networks can get stuck in bad local optima. We examine this behaviour in more detail in the next Section.

Looking at the results from the normalised mismatch kernels (Figure 4.16), we can see that the relative performance among TC-RBM networks is the same, i.e. non-sparse networks have low standard deviation and perform better as we increase the number of hidden units, whereas sparse networks have high standard deviation and their average performance is considerably worse than that of the non-sparse networks.

However, when compared to the rest of the models, the relative performance of the TC-RBM networks is different. More specifically, we can see that the non-sparse network with 10 hidden units is outperformed by most models, including the Dirichlet-Bigram and the empirical marginal distribution. The non-sparse network with 50 hidden units is better than the empirical marginal distribution, but still worse than the Dirichlet-Bigram and only the non-sparse network with 100 hidden units slightly outperforms the Dirichlet-Bigram. We examine why this happens in the next Section, where we visualise the kernel matrices of the mismatch kernels.

4.4.2.1 Visualising the Kernel Matrices

In this Section we try to elucidate some of our previous observations by looking at the mismatch kernel matrices used to compute the squared MMD. Figure 4.17 shows the kernel matrices of the $(4, 1)$ mismatch kernel between testing sequences and samples from different models, where we have truncated very high values to $1,000$, in order to better discern the structure of the kernels. The first three rows (subplots 1 to 9) present kernels matrices corresponding to fold 1, that is for samples drawn from models trained on fold 1. For the non-sparse networks the kernels from other folds are qualitatively similar to the ones presented. On the other hand, for the sparse TC-RBM networks, we have big fluctuations across folds. The performance of sparse networks trained on fold 1 is moderate, close to the overall mean. We additionally present the kernels for the sparse TC-RBM networks trained on fold 9, which gives particularly poor performance. These are presented in the fourth row (subplots 10 to 12) and are denoted by **.

For the sparse networks, and especially for the problematic ones (4th row) we can

see that samples from these models are very similar to each other, while being less similar to the testing sequences, when compared to samples from the non-sparse networks and the Dirichlet-VMM. This abnormality is more prominent as we use less hidden units, suggesting that the constrained use of resources in the sparse networks leads to generative models that assign considerable probability mass to only a few highly prominent configurations.

Comparing the kernel matrices for the non-sparse TC-RBM networks (2nd row) to the kernel matrix for the Dirichlet-VMM, we can see that TCRBM-10 has again an increased overall level of sample-sample similarity, suggesting that a small number of hidden units (10) is not adequate to generate the breadth of different configurations we see in music. On the other hand, when we increase the number of hidden units – TCRBM-50 and TCRBM-100 – the overall level of sample-sample similarity is lower, suggesting that these models assign considerable probability mass to a much larger range of possible configurations. However, we can also notice that the samples from these two networks are less similar to the true testing pieces when compared to samples from the Dirichlet-VMM. In contrast to the Dirichlet-VMM, which effectively parses music into a lexicon of motifs observed in the training data, the TC-RBM learns a distributed representation of melodic features at a motif level. This can lead to TC-RBM samples comprising motifs which are less frequent but still compatible with the melody of the genre. Evaluating this hypothesis would require kernel functions that count the number of unique structures shared between sequences, rather than counting all occurrences of each motif.

The normalised $(4, 1)$ mismatch kernels, depicted in Figure 4.18, exhibit a similar pattern. The sparse TC-RBM networks, and especially the ones in the 4th row, have high sample-sample similarity, while having fairly low sample-test similarity. For the non-sparse networks the overall level of sample-sample similarity is closer to the testing-testing similarity, especially as we use more hidden units. However, when compared to the Dirichlet-VMM we see fewer samples being very similar to testing sequences.

The kernels between training sequences and model samples exhibit similar patterns.

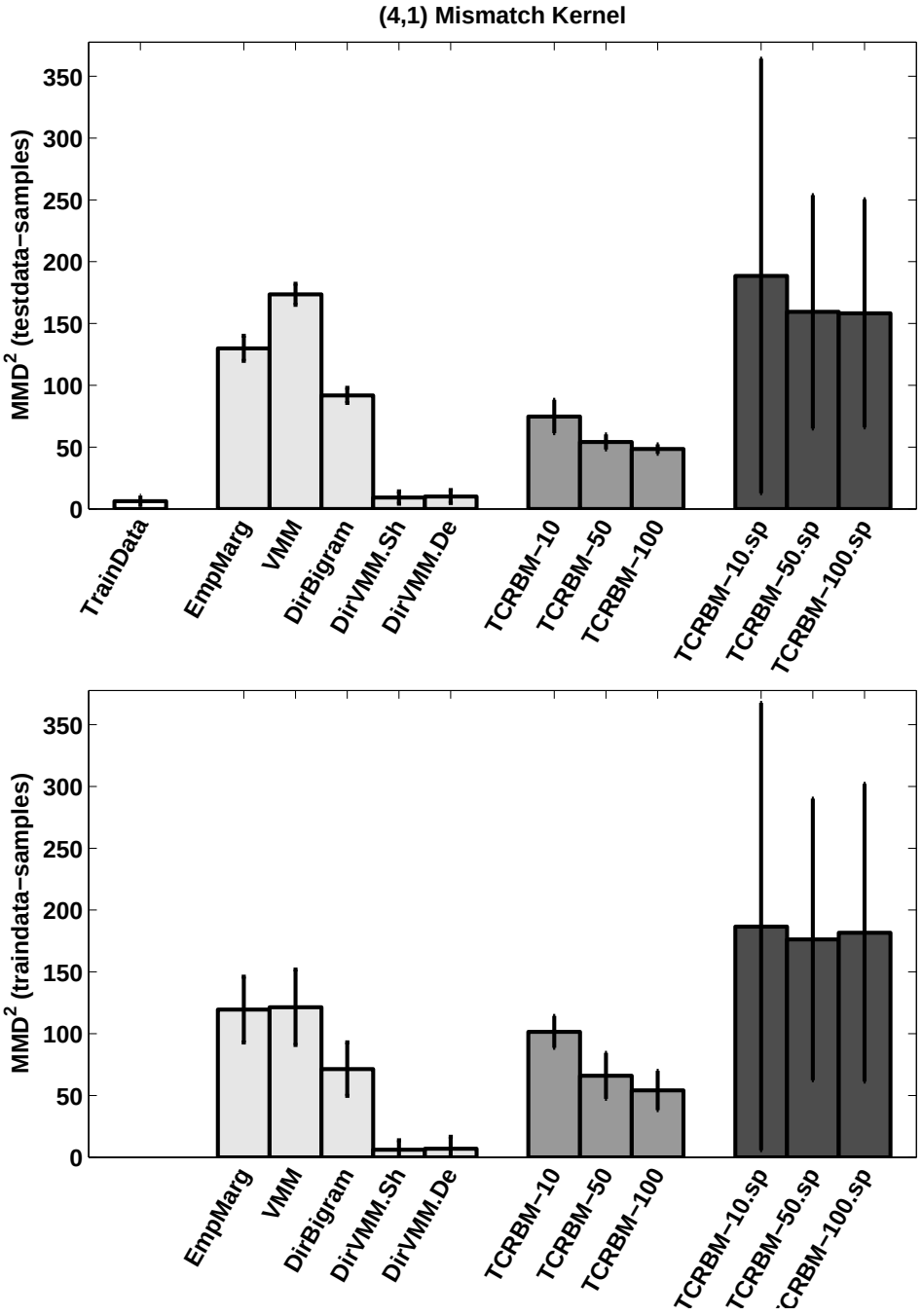


Figure 4.15: Barplots depicting the mean and standard deviation of the estimated squared MMD between data sequences and samples generated from different models. Top: squared MMD between testing sequences and model samples. Bottom: squared MMD between training sequences and model samples. Results using the $(4, 1)$ mismatch kernel.

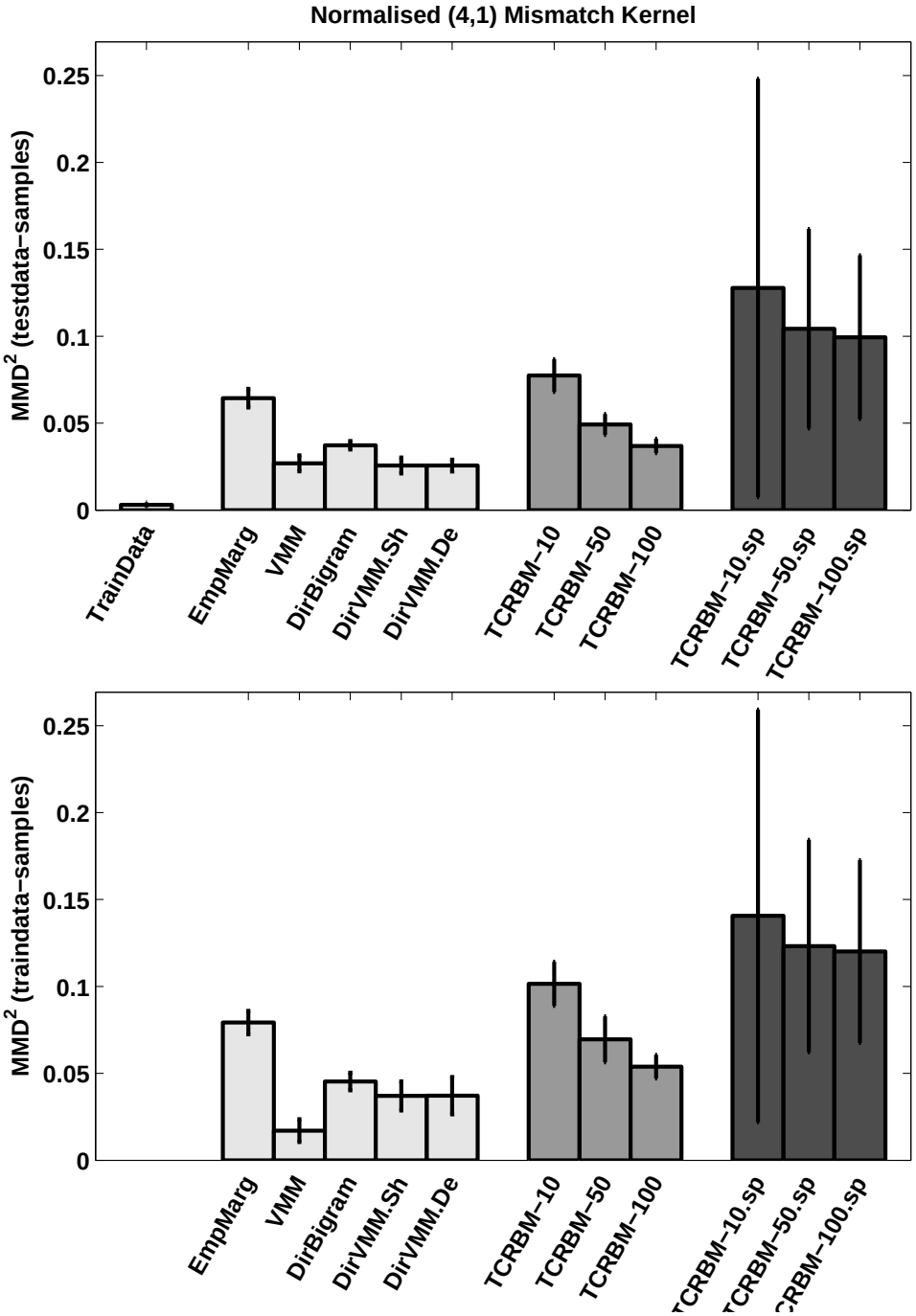


Figure 4.16: Barplots depicting the mean and standard deviation of the estimated squared MMD between data sequences and samples generated from different models. Top: squared MMD between testing sequences and model samples. Bottom: squared MMD between training sequences and model samples. Results using the normalised $(4, 1)$ mismatch kernel.

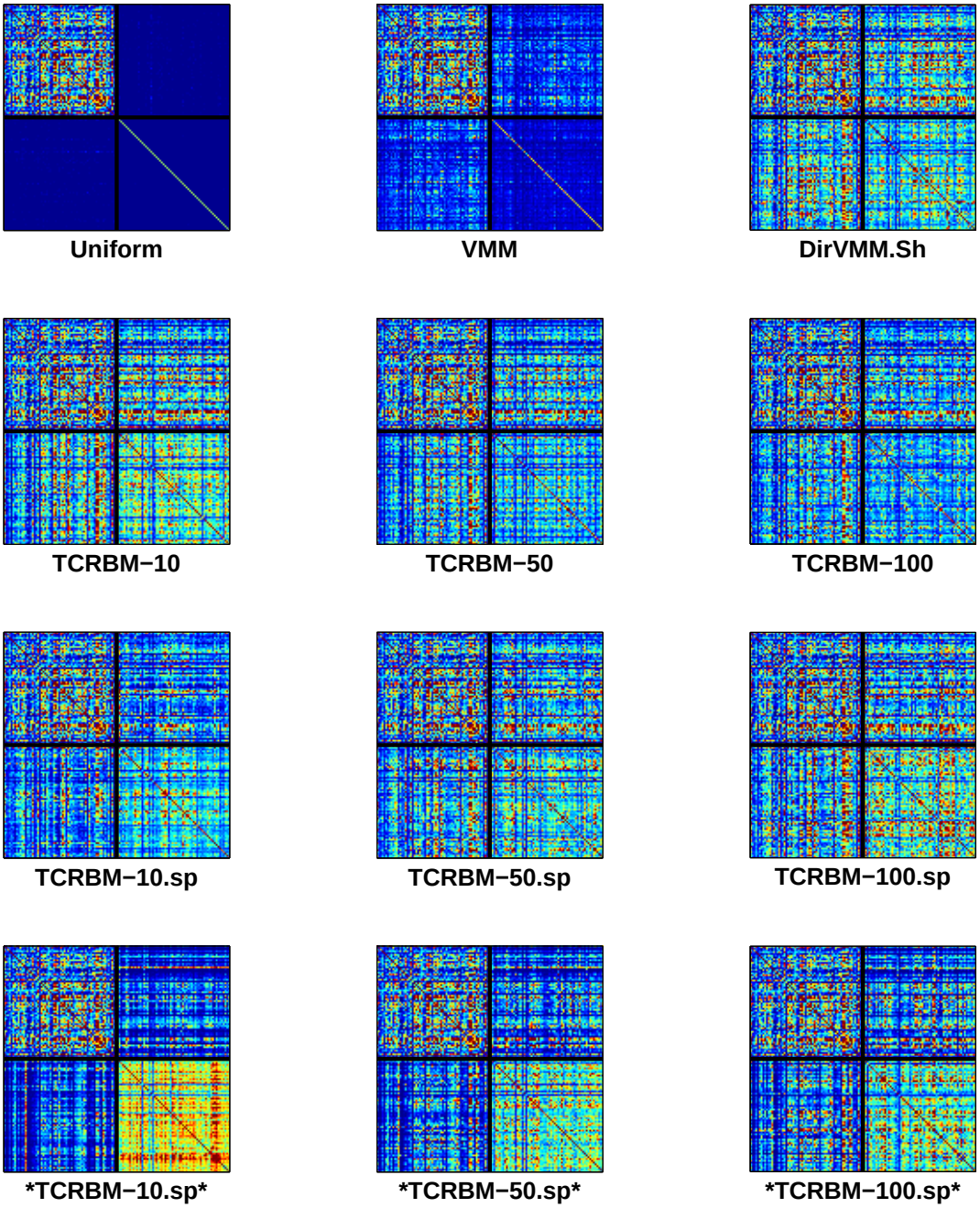


Figure 4.17: The $(4, 1)$ mismatch kernel matrices between testing sequences (rows:1-64) and model samples (rows:65-128) for different models. Colour values range from 0 (dark blue–smallest element across kernels) to $1,000$ (dark red). Values larger than $1,000$ are truncated. The ** denotes results from TC-RBM networks that perform very poorly. See text for details.

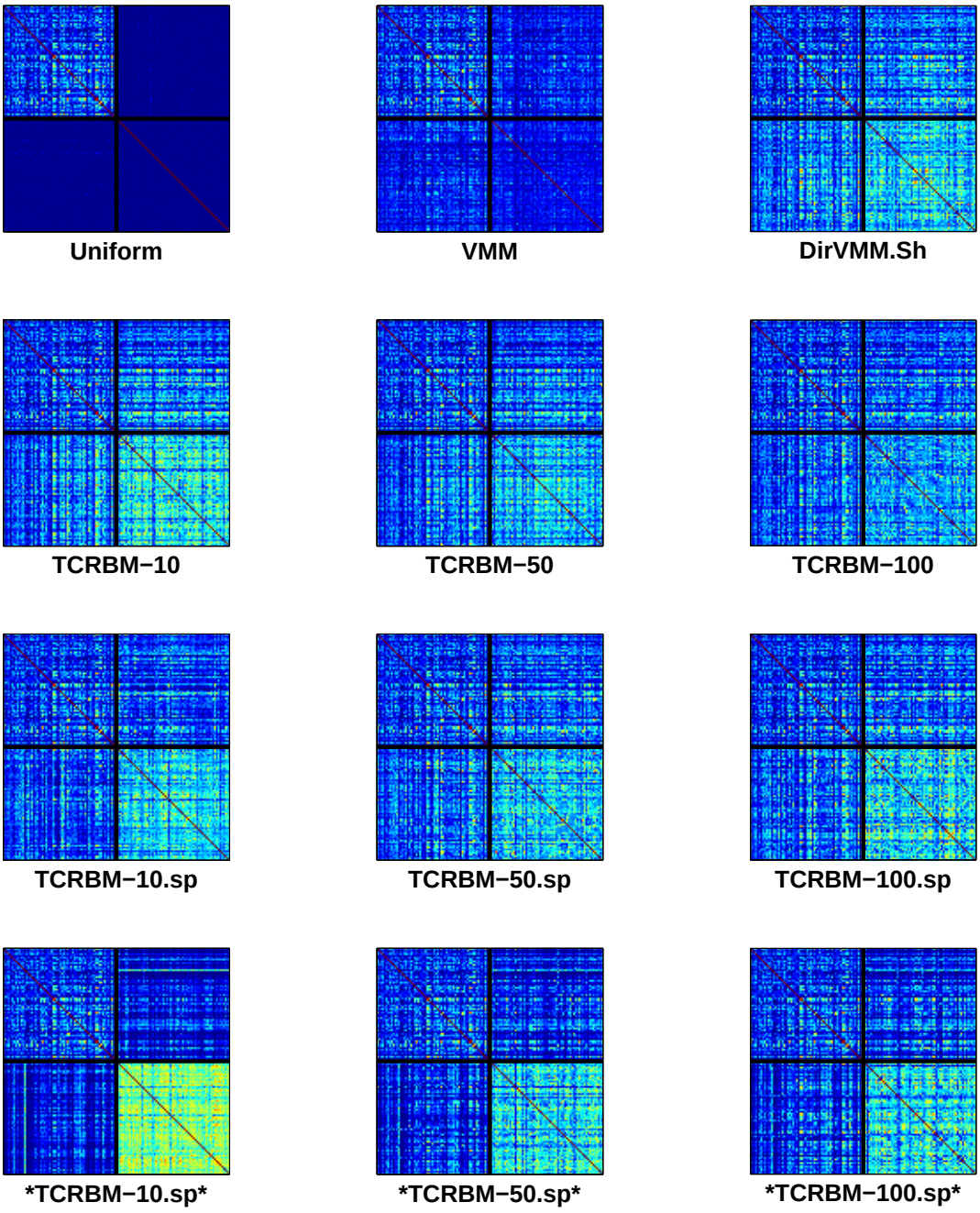


Figure 4.18: The normalised $(4, 1)$ mismatch kernels between testing sequences (rows:1-64) and model samples (rows:65-128) for different models. The ** denotes results from TC-RBM networks that perform very poorly. See text for details.

4.5 Discussion

In this Chapter we addressed the problem of learning a generative model for melody by considering an unsupervised feature extractor, the Time Convolutional Restricted Boltzmann Machine. In contrast to previous RBM approaches for sequential data, the TC-RBM is a fully undirected network that considers visible subsequences, instead of individual time-steps and uses convolution to propagate information through time. Our comparative analysis showed that the Dirichlet-VMM, introduced in Chapter 3, is a better next-step predictor than the TC-RBM. However, we note that the log-likelihood under the Dirichlet-VMM can be computed exactly, whereas in the TC-RBM we employ a Gibbs sampling procedure that approximates the next-step prediction log-likelihood (Algorithm 3). Given that the next-step prediction log-likelihood is effectively the data log-likelihood under a model, it would be interesting to evaluate the TC-RBM using different estimators, such as the ones presented in Murray and Salakhutdinov (2009).

The MMD evaluation revealed that the sparse networks can be bad models for melody generation, with samples being very similar to each other, but not so with true data pieces, suggesting that a lot of the probability mass is allocated to configurations that are unlikely to occur in the data. This is in contrast to the log-likelihood evaluation, where the sparse networks perform comparably to the non-sparse ones. Using a sparse network allows us to interpret what a model has learned and perform qualitative evaluations such as the one in Section 4.3.3, but at the same time it can prevent the network from maximising its use of resources.

Finally, a useful property of the RBM is that it can be stacked to create deep network architectures (Hinton et al., 2006)). Deep models have been shown to learn hierarchical representations of the input space, where higher layers capture successively more abstract features. Interestingly, according to the tonal music theory of Lerdahl and Jackendoff (1983), a listener's musical experience can also be understood as a hierarchy of consecutive abstractions, which can then be used to decompose the structure of a musical piece. In Lerdahl and Jackendoff (1983) every level of the hierarchy is analysed with respect to melody groups, metrical structure, rhythmic structure and rhythmic stability. In the TC-RBM the components or features are learned in an unsupervised manner, and therefore it is difficult to directly associate them with well-defined musical notions. However, it

would be interesting to consider hierarchical approaches for melody generation using a stack of TC-RBMs, where higher layers of musical features could help modulate the appearance of more specific features through time.

Chapter 5

Topic Models

In this Chapter we introduce the Variable-gram Topic model (Spiliopoulou and Storkey, 2012) for the task of melody modelling. Topic models have been primarily used in the natural language processing domain for modelling large collections of documents. The most commonly used topic model is Latent Dirichlet Allocation (LDA) developed by Blei et al. (2003). Given a corpus where each document is seen as a collection of words, LDA discovers a set of abstract latent topics, with each topic being represented by a probability distribution over words. The topics are shared across documents and each document is modelled as a mixture of latent topics, with the mixing proportions being document specific. In this Chapter we will often refer to words and documents to be in accordance with the topic modelling formalism. In the context of music modelling these correspond to note symbols – pitch including pitch class, silence and continuation – and music pieces, respectively.

The models we examined in the previous chapters, namely the Dirichlet-VMM and the TC-RBM, are capable of capturing the statistical dependencies of a musical genre as a whole. However, being stationary models, they are not able to distinguish between the inter and intra-piece structure in melodies. Musical pieces from the same genre are built using the same musical form, and so we can expect structural elements, for instance motifs, to be shared across pieces. At the same time, within a piece, musical phrases are altered and recombined aiming to elaborate the central idea of the piece. Hence, we might expect that each piece uses only a subset of the available structural elements (e.g. motifs)

and that these elements are likely to be re-used within the same piece. A topic model can represent this difference between inter and intra-sequence structure by allowing each musical piece to have a unique probability distribution over latent topics. The more dissimilar the distributions of different pieces over topics, the more we can expect the inter and intra-piece similarity to differ, provided that the latent topics are also different from each other.

The use of topic models for music is also motivated by the fact that within a single piece of music we can have phrases with substantially different style or dynamics. In Western music a piece often comprises consonant and dissonant parts, with the dissonant parts creating tension, which is then resolved by consonant ones. The pace of music is also likely to change within a piece, giving rise to parts that are fast moving and parts that are slow. A topic model can represent these different music regimes as abstract latent topics, each of which is associated with different probability distributions over words (notes). The reels dataset has relatively simple structure and does not exhibit substantial pace or dissonant/consonant variation. However, we find that the latent topics are able to distinguish pieces from different key signatures – remember that our dataset comprises untransposed pieces from the key of D and the key of G – and are also able to capture low-level musical elements such as note duration.

The Variable-gram Topic model couples the latent topic formalism with a systematic model of contextual information. It employs the Dirichlet-VMM, introduced in Chapter 3, for the parametrisation of the topic distributions over words. Therefore, in contrast to LDA, where each topic is represented by a single probability distribution over words, in the Variable-gram Topic model each topic is represented by a set of conditional probability distributions, each representing the belief of a Bayesian agent from the Dirichlet-VMM. This allows the model to capture local dependencies by learning variable-order contexts indicative of the future, while using the latent topics to represent different music regimes, for instance, allowing motifs to have different function (be associated with different probability distributions) depending on the key signature.

We begin with a short description of Latent Dirichlet Allocation; move on to detailing the Variable-gram Topic model and how inference and learning in the model is performed; and finally examine its performance on the melody modelling task.

5.1 Latent Dirichlet Allocation

As already mentioned, Latent Dirichlet Allocation is a statistical model for document collections. The main assumption is that each document addresses a particular set of topics and thus each word in a document can be expressed as a probabilistic mixture of these topics. Therefore, if we represent topics by a discrete variable z and we have K topics the probability of the t-th word in document d, $x_{t,d}$ will be given by

$$P(x_{t,d}) = \sum_{k=1}^{K} P(x_{t,d}|z_{t,d} = k)P(z_{t,d} = k) \ . \tag{5.1}$$

The first term in the RHS, $P(x_{t,d}|z_{t,d} = k)$, is the probability of the word under the k-th topic, and shows how often each word occurs when we write about a certain topic. For instance, if the topic is "sports" we can expect the word "athlete" to appear more frequently than the word "singer". The second term, $P(z_{t,d} = k)$, is the probability of selecting topic k for the t-th word of document d and corresponds to the mixing proportions of the topics.

LDA assumes that the mixing proportions for the topics are document specific, so that for each document, $P(z_{t,d})$ is given by a categorical distribution with parameter vector $\boldsymbol{\theta}_d$, so that $P(z_{t,d} = k) = \theta_{k|d}$, $k \in \{1, \ldots, K\}$ and $\sum_k \theta_{k|d} = 1$. It further assumes that there is a prior distribution over these parameters vectors, which is chosen to be a Dirichlet distribution, mostly due to its conjugacy to the categorical.

Additionally, as seen from (5.1) each topic, k, is associated with a probability distribution over words. In LDA, we assume that words come from a finite alphabet (vocabulary) Σ and thus for each topic k, $P(x_{t,d}|z_{t,d} = k)$ is a categorical distribution parametrised by a discrete vector $\boldsymbol{\phi}_k$, so that $P(x_{t,d} = i|z_{t,d} = k) = \phi_{i|k}$, $i \in \{1, \ldots \|\Sigma\|\}$ and $\sum_i \phi_{i|k} = 1$. In the original work by Blei et al. (2003), the $\boldsymbol{\phi}_k$'s were approximated using variational Bayes. However, Griffiths and Steyvers (2004) introduced a prior Dirichlet distribution over the $\boldsymbol{\phi}_k$'s, which allowed them to perform inference using a Markov chain Monte Carlo algorithm. This inference procedure is faster and has been shown to perform better, which is why the model with both priors is typically used.

The complete model with both priors is given by

$$\begin{aligned}
\boldsymbol{\theta}_d &\sim \text{Dirichlet}(\alpha\mathbf{n}) \\
z_{t,d}|\boldsymbol{\theta}_d &\sim \text{Categorical}(\boldsymbol{\theta}_d) \\
\boldsymbol{\phi}_k &\sim \text{Dirichlet}(\beta\mathbf{m}) \\
x_{t,d}|z_{t,d}, \boldsymbol{\phi}_{z_{t,d}} &\sim \text{Categorical}(\boldsymbol{\phi}_{z_{t,d}}) \ ,
\end{aligned}$$

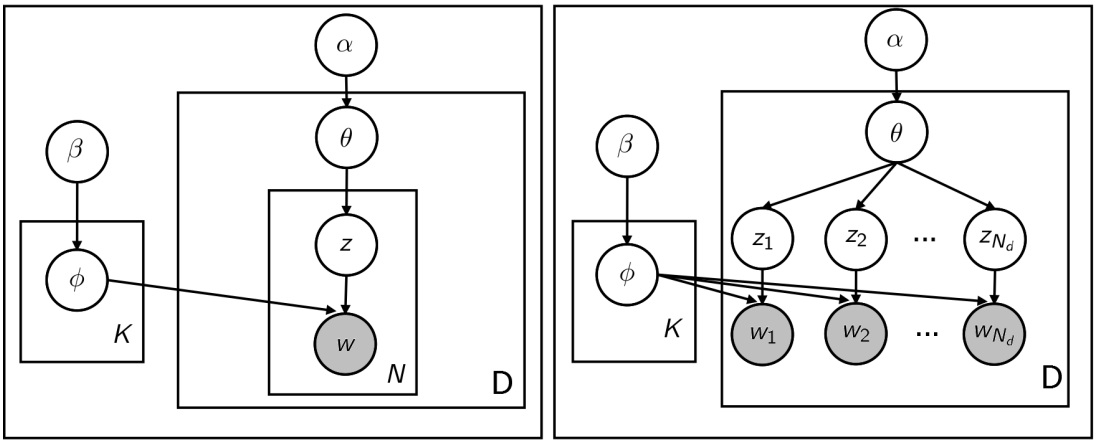
where we represent the hyperparameters of the Dirichlet distributions using a product between a concentration parameter, α and β, and a base measure, $\mathbf{n}$ and $\mathbf{m}$, respectively.

The graphical model for Latent Dirichlet Allocation is shown in Figure 5.1(a). Note that LDA assumes that both documents in a corpus and words in a document are exchangeable, i.e. the order in which these observations occur is neglected. Word exchangeability is related to the bag-of-words assumption, which is often used in natural language processing systems. In a bag-of-words model, a document is represented as an unordered collection of words and the observation of interest is whether a word has occurred and/or how often it occurs.

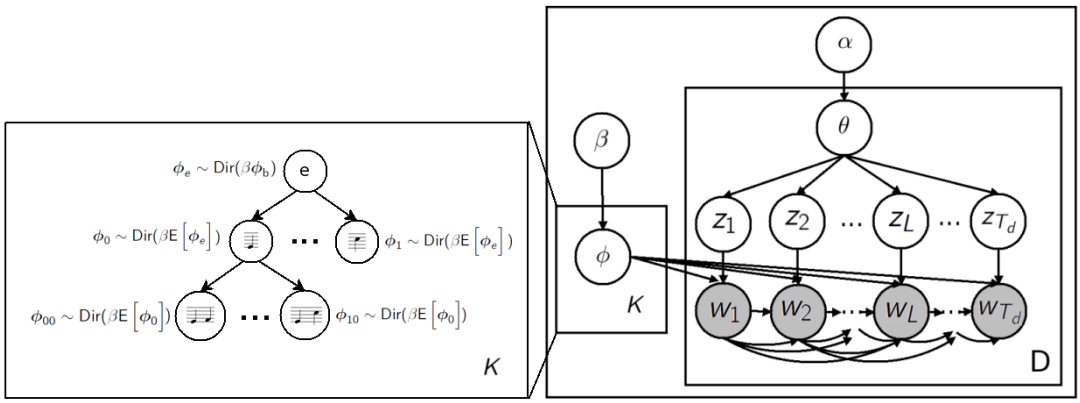
5.2 The Variable-gram Topic Model

The graphical model for the Variable-gram Topic model is depicted in Figure 5.1(b). Similar to LDA, each document is modelled as a mixture of latent topics and the latent topics are shared among documents in a corpus. On the other hand, each topic is now defined through a Dirichlet-VMM; instead of a single probability distribution over words, we now have a set of conditional probability distributions encoding contextual information of variable order. This difference from LDA is apparent in Figure 5.1, where we can see that in the Variable-gram Topic model, each word receives directed connections from the corresponding latent topic and the L previously observed words. The latter connections are defined in terms of the Dirichlet-VMM, which means that depending on the context we observe, we can have $\ell, \ell \in \{1, \ldots, L\}$, active connections. Note that if we consider only first order ($\ell = L = 1$) dependencies, instead of variable order ones, then we retrieve the Bigram Topic model of Wallach (2006).

The motivation for using the Dirichlet-VMM to model the word distributions



(a)



(b)

Figure 5.1: (a) Graphical model for Latent Dirichlet Allocation with the plate notation for words (left) and without (right). (b) Graphical model for the Variable-gram Topic Model.

comes from the fact that the bag-of-words assumption is quite restrictive for modelling music; the order in which events occur carries important information about the structure of the data. In fact, this is also a common criticism for topic models when they are used directly for modelling language.

Let $\mathbf{X} = \{\mathbf{x}^1, \ldots, \mathbf{x}^D\}$ be a corpus of D documents, with $\mathbf{x}^d = \{x_{1,d}, \ldots, x_{T_d,d}\}$, where $d \in \{1, \ldots, D\}$ indexes documents and T_d is the length of document d. We assume that words come from a finite alphabet (vocabulary) Σ, so that $x_{t,d} \in \{1, \ldots, \|\Sigma\|\}$, where $t \in \{1, \ldots, T_d\}$ indexes word tokens in document d. Now, consider a Dirichlet-VMM tree, $\mathcal{T}$, and let J be the number of leaf nodes in the tree, each associated with a context $\mathbf{c}_j$, $j \in \{1, \ldots, J\}$. Remember that in

Algorithm 4 Generative Process for the Variable-gram Topic Model.

Input: α, $\mathbf{n}$, β, $\{\mathbf{m}_{j,k}\}$, Dirichlet-VMM tree $\mathcal{T}$
for each topic $k, k \in \{1, \ldots, K\}$ **do**
  for each context $j, j \in \{1, \ldots, J\}$, $J =$ (#leaf nodes in $\mathcal{T}$) **do**
    Draw $\boldsymbol{\phi}_{j,k} \sim \text{Dirichlet}(\beta \mathbf{m}_{j,k}))$
  end for
end for
for each document $d, d \in \{1, \ldots, D\}$ **do**
  Draw $\boldsymbol{\theta}_d \sim \text{Dirichlet}(\alpha \mathbf{n})$
  for each token $t, t \in \{1, ..., T_d\}$ in d **do**
    Choose a topic $z_{t,d} \sim \text{Categorical}(\boldsymbol{\theta}_d)$
    Choose a word $x_{t,d} \sim \text{Categorical}(\boldsymbol{\phi}_{\mathbf{c}_{t,d}, z_{t,d}})$
  end for
end for

the Dirichlet-VMM only the leaf nodes are used for prediction, that is to get the conditional probability distribution of the next symbol given an observed context, we start at the root node and traverse down the tree reading off the context from the most recent observation to the oldest one, until we reach a leaf node. Finally, let K be the number of available topics, indexed by $k \in \{1, \ldots, K\}$.

Under the Variable-gram Topic model, each document, d, has a distribution over the K latent topics parametrised by $\boldsymbol{\theta}_d$, with $\theta_{k|d} = P(z_{t,d} = k)$. Similar to LDA, the $\boldsymbol{\theta}_d$'s are drawn from a prior Dirichlet distribution, with concentration parameter α and base measure $\mathbf{n}$. Overall, topic generation is defined by

$$\boldsymbol{\theta}_d \sim \text{Dirichlet}(\alpha \mathbf{n}) \tag{5.2}$$

$$z_{t,d} | \boldsymbol{\theta}_d \sim \text{Categorical}(\boldsymbol{\theta}_d) \ . \tag{5.3}$$

Each topic is associated with a Dirichlet-VMM, so that word generation within a topic, k, is defined by the set of conditional distributions identifying the leaf nodes of the Dirichlet-VMM tree, $\mathcal{T}$. The structure of the tree is the same for all the topics, so that the set of available contexts is fixed. However, the conditional distributions in the tree are now topic–specific. Word generation is parametrised by $\boldsymbol{\phi}_{j,k}$, with $\phi_{i|j,k} = P(x_{t,d} = i | \mathbf{c}_{t,d} = \mathbf{c}_j, z_{t,d,} = k)$. Each topic is now represented by a matrix $\Phi_k = \left[\boldsymbol{\phi}_{1,k} \ldots \boldsymbol{\phi}_{J,k}\right]$, where the j-th column is the

conditional probability distribution over the word, given context $\mathbf{c}_j$. From the viewpoint of the Dirichlet-VMM, we can think of the Bayesian agents as being topic–specific.

The prior distribution of the $\boldsymbol{\phi}_{j,k}$'s is also defined through the Dirichlet-VMM. According to the model definition in (3.5), the prior distribution for the Bayesian agent at node j is a Dirichlet distribution with concentration parameter β and base measure given by the posterior predictive distribution at the parent node. Let $\mathbf{m}_{j,k} = \hat{\boldsymbol{\phi}}_{\mathbf{pa}(j),k}$ be the base measure for leaf node j defined by the Dirichlet-VMM of topic k. Then word generation is given by

$$\boldsymbol{\phi}_{j,k} \sim \text{Dirichlet}(\beta \mathbf{m}_{j,k}) \tag{5.4}$$

$$x_{t,d} | z_{t,d}, \mathbf{c}_{t,d}, \boldsymbol{\phi}_{\mathbf{c}_{t,d}, z_{t,d}} \sim \text{Categorical}(\boldsymbol{\phi}_{\mathbf{c}_{t,d}, z_{t,d}}) \ . \tag{5.5}$$

Overall, topic generation is parametrised by a matrix, $\boldsymbol{\Theta}$, with $D(K-1)$ free parameters, and word generation is parametrised by a tensor, $\boldsymbol{\Phi}$, with $KJ(\|\Sigma\|-1)$ free parameters. The generative process for a corpus $\mathbf{X}$ given hyperparameters α, $\mathbf{n}$, β, a Dirichlet-VMM tree, $\mathcal{T}$, and an estimate of the $\mathbf{m}_{j,k}$'s is given in Algorithm 4.

Let $N_{i|j,k}$ be the number of times word i has been assigned to topic k when preceded by context $\mathbf{c}_j$ and $N_{k|d}$ the number of times topic k has occurred in document d. Then using the definition of the categorical distribution, the joint probability of a corpus $\mathbf{X}$ and a set of corresponding latent topic assignments $\mathbf{Z}$ under a Variable-gram Topic model with parameters $\boldsymbol{\Theta}$ and $\boldsymbol{\Phi}$ is given by

$$\begin{aligned} P(\mathbf{X}, \mathbf{Z} | \boldsymbol{\Phi}, \boldsymbol{\Theta}) &= \prod_{d=1}^{D} \prod_{t=1}^{T_d} P(z_{t,d} | \boldsymbol{\Phi}, \boldsymbol{\Theta}) P(x_{t,d} | \mathbf{c}_{t,d}, z_{t,d}, \boldsymbol{\Phi}, \boldsymbol{\Theta}) \\ &= \prod_{d=1}^{D} \prod_{t=1}^{T_d} P(z_{t,d} | \boldsymbol{\theta}_d) P(x_{t,d} | \boldsymbol{\phi}_{\mathbf{c}_{t,d}, z_{t,d}}) \\ &= \prod_{d=1}^{D} \prod_{k=1}^{K} \left(\theta_{k|d}^{N_{k|d}} \prod_{j=1}^{J} \prod_{i=1}^{\|\Sigma\|} \phi_{i|j,k}^{N_{i|j,k}} \right) \ . \end{aligned} \tag{5.6}$$

From (5.6) we can see that conditioned on the model parameters, the joint probability of the complete data has a simple, factored form expressed using the frequency counts of topics within a document and of word-context pairs within a topic.

5.2.1 Inference and Parameter Estimation

To estimate the model parameters, $\mathbf{\Phi}$ and $\mathbf{\Theta}$, we use the approach of Griffiths and Steyvers (2004) and consider the posterior distribution over latent topics, $P(\mathbf{Z}|\mathbf{X})$, and a Markov chain Monte Carlo algorithm to get samples from this posterior distribution. Each state of the Markov chain is an assignment of values to all latent topic variables, $\mathbf{Z}$. To transition between states, we use a Rao-Blackwellised collapsed Gibbs sampler, where we sequentially sample all latent topics from their distribution, conditioned on the data and the current value of all other latent topics, having marginalised out the model parameters, $\mathbf{\Phi}$ and $\mathbf{\Theta}$. After the Markov chain has converged, we get the Rao-Blackwell estimates for $\mathbf{\Phi}$ and $\mathbf{\Theta}$ by examining the posterior distribution over latent topics. Note that we only check for convergence by visually examining the topic allocations. The inference procedure is detailed in Appendix A. Here we introduce key results and give an overview of the algorithm.

Using (5.6) and the model definition for the prior distributions over the parameters, the total probability of the model given a set of hyperparameters, α, $\mathbf{n}$, β, and an estimate of the $\{\mathbf{m}_{j,k}\}$'s is given by

$$P(\mathbf{X}, \mathbf{Z}, \mathbf{\Phi}, \Theta|\alpha, \mathbf{n}, \beta, \{\mathbf{m}_{j,k}\}) =$$
$$\prod_{d=1}^{D} P(\boldsymbol{\theta}_d|\alpha\mathbf{n}) \prod_{k=1}^{K}\prod_{j=1}^{J} P(\boldsymbol{\phi}_{j,k}|\beta\mathbf{m}_{j,k}) \prod_{d=1}^{D}\prod_{t=1}^{T_d} P(z_{t,d}|\boldsymbol{\theta}_d) P(x_{t,d}|\boldsymbol{\phi}_{\mathbf{c}_{t,d},z_{t,d}}) \ . \qquad (5.7)$$

From (5.7) we can see that, likewise LDA, given the hyperparameters $\alpha\mathbf{n}$, the $\boldsymbol{\theta}_d$ parameters are independent from each other and the same for all the $\boldsymbol{\phi}_{j,k}$'s. Similarly, the $\boldsymbol{\phi}_{j,k}$ parameters are independent from each other given $\beta\{\mathbf{m}_{j,k}\}$. Using these independence relations and the conjugacy of the Dirichlet to the categorical distribution, we can integrate over the model parameters, $\mathbf{\Theta}$ and $\mathbf{\Phi}$, in (5.7), to obtain a closed form solution for the joint probability of a corpus $\mathbf{X}$ and a set of corresponding latent topic assignments $\mathbf{Z}$, given a set of hyperparameters.

The complete derivation is given in Section A.2. We have

$$P(\mathbf{Z}, \mathbf{X}|\alpha\mathbf{n}, \beta\{\mathbf{m}_{j,k}\}) = \tag{5.8}$$

$$\prod_{d=1}^{D} \frac{\Gamma(\alpha)}{\prod_{k=1}^{K}\Gamma(\alpha n_k)} \frac{\prod_{k=1}^{K}\Gamma(\alpha n_k + N_{k|d})}{\Gamma(\alpha + N_d)} \tag{5.9}$$

$$\times \prod_{k=1}^{K}\prod_{j=1}^{J} \frac{\Gamma(\beta)}{\prod_{i=1}^{\|\Sigma\|}\Gamma(\beta m_{i|j,k})} \frac{\prod_{i=1}^{\|\Sigma\|}\Gamma(\beta m_{i|j,k} + N_{i|j,k})}{\Gamma(\beta + N_{j,k})} \ . \tag{5.10}$$

Using (5.10) we can define a collapsed Gibbs sampling procedure, where we only sample the latent topic assignments and keep track of the model parameters $\mathbf{\Phi}$ and $\mathbf{\Theta}$ through their sufficient statistics. The procedure starts by randomly initialising the latent topic assignments, and then sequentially sampling each latent variable $z_{t,d}$ given the current values of all other latent variables $\mathbf{Z}_{-(t,d)}$, the data $\mathbf{X}$ and a set of hyperparameters $\{\alpha\mathbf{n}, \beta\{\mathbf{m}_{j,k}\}\}$. At every Gibbs step we sample $z_{t,d}$ according to (see Section A.3 for a derivation)

$$P(z_{t,d} = k|x_{t,d} = i, \mathbf{c}_{t,d} = \mathbf{c}_j, \mathbf{Z}_{-(t,d)}, \mathbf{X}_{-(t,d)}, \alpha\mathbf{n}, \beta\{\mathbf{m}_{j,k}\}) \propto$$
$$\frac{\alpha n_k + \left\{N_{k|d}\right\}_{-(t,d)}}{\alpha + \left\{N_d\right\}_{-(t,d)}} \times \frac{\beta m_{i|j,k} + \left\{N_{i|j,k}\right\}_{-(t,d)}}{\beta + \left\{N_{j,k}\right\}_{-(t,d)}} \ , \tag{5.11}$$

where $N_d = T_d$ is the number of tokens in document d, $N_{j,k}$ is the number of times topic k has been assigned to any word preceded by context $\mathbf{c}_j$ and subscript $-(t,d)$ denotes that counts associated with the t-th token of document d are excluded.

Therefore, Gibbs sampling proceeds, by sequentially subtracting the current word and topic from the counts associated with $z_{t,d}^{\text{old}}$, sampling a new topic using (5.11), and adding the new word and topic to the counts associated with $z_{t,d}^{\text{new}}$.

Every time we subtract or add counts, we also update the Dirichlet-VMM trees of the old and the new topic, respectively. In the Dirichlet-VMM the counts "sit" on every node whose conditional probability distribution satisfies the data. So, for a given word-context pair, $(x, \mathbf{c})$, we need to update the counts at all nodes in the path from the leaf node with context $\mathbf{c}$ to the root. Note that these nodes have contexts of the form suffix($\mathbf{c}$). In order to reduce computation we do not perform the complete iterative inference procedure of the Dirichlet-VMM every time the counts change. Instead, we keep track of the counts at every node and estimate each $\mathbf{m}_{j,k}$ only when needed by the Gibbs sampler.

For any state of the Markov chain $\mathbf{Z}^{(s)}$, we can get an estimate for $\mathbf{\Phi}$ and $\mathbf{\Theta}$ through the posterior predictive distributions over words and topics, respectively. We have

$$\hat{\phi}_{i|j,k} = P(x = i|\mathbf{c} = j, z = k, \mathbf{X}, \mathbf{Z}^{(s)}, \beta\mathbf{m}_{j,k}^{(s)}) = \frac{\beta m_{i|j,k}^{(s)} + N_{i|j,k}^{(s)}}{\beta + N_{j,k}^{(s)}} \ , \tag{5.12}$$

$$\hat{\theta}_{k|d} = P(z = k|\mathbf{X}, \mathbf{Z}^{(s)}, \alpha\mathbf{n}) = \frac{\alpha n_k + N_{k|d}^{(s)}}{\alpha + N_d^{(s)}} \ . \tag{5.13}$$

5.3 Model Evaluation

For the purposes of this work we set the hyperparameters α and β through cross-validation and we use symmetric Dirichlet distributions as priors. More specifically, for the prior distribution over topics we have $\mathbf{n} = [n_1 \dots n_K]^\mathsf{T}$, with $n_1 = \dots = n_K = \frac{1}{K}$ and $\sum_{k=1}^{K} n_k = 1$. Wallach et al. (2009a) finds that an asymmetric Dirichlet distribution is a better prior over $\mathbf{\Theta}$ when modelling natural language documents, where we have highly skewed word frequency distributions. In the context of melody, the vocabulary size is small and therefore we do not observe a power-law in the frequency of the symbols. We note, however, that it would be interesting to examine the behaviour of the variable-gram topic model with asymmetric priors over the document-topic distributions.

The prior distributions over words are defined through the inference procedure of the Dirichlet-VMM. We use a symmetric Dirichlet distribution at the root node of every tree, so that the base measure for the prior at the root node is given by $\boldsymbol{\phi}_{b,k} = \left[\phi_{1|b,k} \dots \phi_{\|\Sigma\| \ |b,k}\right]^\mathsf{T}$, with $\phi_{1|b,k} = \dots = \phi_{\|\Sigma\| \ |b,k} = \frac{1}{\|\Sigma\|}$ and $\sum_{i=1}^{\|\Sigma\|} \phi_{i|b,k} = 1$, $\forall k \in \{1, \dots, K\}$.

In this Section we evaluate the performance of topic models with respect to different settings of the hyperparameters α and β. With respect to the contextual information being considered, we examine four different models; latent Dirichlet allocation, which does not consider any contextual information, the Bigram Topic model (Wallach, 2006), which considers first order ($\ell = L = 1$) dependencies, and two Variable-gram Topic models, each with a different tree structure for the Dirichlet-VMM. The two trees are obtained by changing the threshold that the relative frequency of a context must exceed, in order to be included in the tree. The first tree is relatively shallow (threshold: $\epsilon_1 = 10^{-3}$) and is referred to as $.Sh$,

whereas the second tree is deeper (threshold: $\epsilon_1 = 10^{-4}$) and is referred to by *.De*. Finally, for each topic model we present results using 5, 10 and 50 topics.

The quantity we are interested in is the average prediction log-likelihood of held-out data under a model. Let $\mathbf{X}$ denote a set of unseen documents and consider a model $\mathcal{M}$ with learned parameters $\mathbf{u}$. The average prediction log-likelihood is given by

$$\mathcal{L}(\mathbf{u}, \mathcal{M}; \mathbf{X}) = \frac{1}{\sum_{d=1}^{D} T_d} \sum_{d=1}^{D} \sum_{t=1}^{T_d} \log P_{\mathcal{M}}(x_{t,d} | x_{t-1,d}, \ldots, x_{0,d}, \mathbf{u}) \ , \tag{5.14}$$

where $x_{0,d}$ is a null observation specifying the absence of events at the beginning of a document. It is easy to see that this is also the average log-probability that a model assigns to a token from an unseen document. Therefore, for topic models computing (5.14) is equivalent to evaluating

$$\begin{aligned} P(\mathbf{X}|\mathbf{\Phi}, \alpha\mathbf{n}) &= \prod_d P(\mathbf{x}_d|\mathbf{\Phi}, \alpha\mathbf{n}) \\ &= \prod_d \sum_{\mathbf{z}_d} P(\mathbf{x}_d, \mathbf{z}_d|\mathbf{\Phi}, \alpha\mathbf{n}) \ , \end{aligned} \tag{5.15}$$

where we have dropped the dependency on the hyperparameters for the word prior, β and $\mathbf{m}$ or $\{\mathbf{m}_{j,k}\}$, as we are given the learned model parameters $\mathbf{\Phi}$. Note that for an unseen document, d, the parameter vector for topic generation, $\boldsymbol{\theta}_d$, is unknown. Exact computation of this expression is intractable for realistic problems, as to compute the likelihood of each document, d, we need to sum over the K^{T_d} states that $\mathbf{z}_d$ can take.

A few different methods have been proposed for estimating the probability of unseen data under a topic model, with much of the earlier literature on topic models using the harmonic mean (e.g. Griffiths and Steyvers (2004) and Wallach (2006)). We refer the reader to Wallach et al. (2009b) for an evaluation of different methods for estimating (5.15). Here we use a "left-to-right" method, which was first proposed by Wallach (2008) and further examined in Wallach et al. (2009b), where it is found to give a more accurate estimation of (5.15) compared to several commonly used alternatives.

The "left-to-right" evaluation algorithm casts the estimation of the likelihood of an unseen document as a next-step prediction problem. Let $\mathbf{x}$ be an unseen

document of length T, and subscript $< t$ index variables associated with tokens that precede token t. We have

$$P(\mathbf{x}|\mathbf{\Phi}, \alpha\mathbf{n}) = \prod_{t=1}^{T} P(x_t|\mathbf{x}_{<t}, \mathbf{\Phi}, \alpha\mathbf{n}) = \prod_{t=1}^{T} \sum_{\mathbf{z}_{\leq t}} P(x_t, \mathbf{z}_{\leq t}|\mathbf{x}_{<t}, \mathbf{\Phi}, \alpha\mathbf{n})$$
$$= \prod_{t=1}^{T} \sum_{\mathbf{z}_{\leq t}} P(x_t|z_t, \mathbf{z}_{<t}, \mathbf{x}_{<t}, \mathbf{\Phi}, \alpha\mathbf{n}) P(z_t|\mathbf{z}_{<t}, \mathbf{x}_{<t}, \mathbf{\Phi}, \alpha\mathbf{n}) P(\mathbf{z}_{<t}|\mathbf{x}_{<t}, \mathbf{\Phi}, \alpha\mathbf{n})$$
$$= \prod_{t=1}^{T} \sum_{z_t} P(x_t|\boldsymbol{\phi}_{\mathbf{c}_t, z_t}) \sum_{\mathbf{z}_{<t}} P(z_t|\mathbf{z}_{<t}, \mathbf{x}_{<t}, \mathbf{\Phi}, \alpha\mathbf{n}) P(\mathbf{z}_{<t}|\mathbf{x}_{<t}, \mathbf{\Phi}, \alpha\mathbf{n}) \ . \qquad (5.16)$$

To estimate (5.16) we can consider the following sequential sampling scheme. At the first time step – no preceding observations – we sample S topics from the prior over topic distributions. Then for each of these samples we update the posterior distribution over the topics with the sampled assignment $z_1^{(s)}$. However, these updates result in different posterior distributions and thus at the next time-step we need to sample S^2 topics, that is S topics from each posterior, and so on. Here, we use an approximation to this sampling procedure, where at each time-step we keep a single posterior distribution over topics, which only considers preceding tokens, we compute the likelihood of the time-step by taking an expectation over the topic posterior, and finally, update the posterior distribution, by taking the mean of the posterior distribution after observing the current time-step.

Additionally we use four update schemes for the distributions over words. In the first approach, denoted by $S.1$, we do not update the word distribution during testing, that is the information from the observed part of a held-out piece is only used to update the parameters for topic generation. In the second approach, denoted by $S.2$, after each time-step we also update the distributions over words by sampling a topic, k, from our approximation of the posterior distribution and add a count to the $\boldsymbol{\phi}_{i|j,k}$ corresponding to the observed word-context. In the third approach, $S.3$, the distributions over words are updated by adding to the $\boldsymbol{\phi}_{i|j,k}$'s for all K topics, counts corresponding to the posterior probability of each topic k. Finally, in approach $S.4$ we update the $\boldsymbol{\phi}_{i|j,k}$'s for all topics by adding a count to the observed word-context.

Looking at the results for Latent Dirichlet Allocation (Table 5.1) we can see that, in most cases, the best setting for the concentration parameters of the prior distributions is $\alpha = 5$ and $\beta = 1$. The model is more sensitive to the value of the

α hyperparameter, which controls the distribution over topic assignments. Given that the prior is a symmetric Dirichlet distribution, a small α parameter leads to a sharper posterior distribution over topics, which means that each document (music piece) is more likely to be represented by a small number of topics. On the other hand, a large α value places a lot of importance on the prior uniform measure, so that the posterior distribution spreads its mass across all topics in every document. The value of 5 for the concentration parameter over topics gives the highest log-likelihood, although a value of 1 and a value of 10 give rise to similar results.

LDA is less sensitive to the value for the β hyperparameter, that is the concentration parameter of the prior Dirichlet distribution over words for each latent topic. We can see that for a given α, different values for the β parameter give similar results. However, we can note that although the differences are only marginal, smaller values are consistently better, suggesting that topics with sharper posterior distributions over words (notes) are preferable.

With respect to the number of topics, we can see that with update scheme $S.1$, performance is best when we use 10 latent topics. Increasing the number of topics from 5 to 10 gives higher log-likelihood, but a further increase to 50 topics is not beneficial. Remember that in update scheme $S.1$, we do not update the word distributions during testing. This suggests that with 50 topics the model starts to overfit the training data, so that the distributions over words can no longer generalise well to unseen documents. This is also the case when we use update scheme $S.3$, where a topic's distribution over words is updated after each time-step with a normalised count, proportional to the posterior probability of the topic. In this case, the information coming from the held-out data, is not enough to overcome the problem of overfitting.

On the other hand, when we use update schemes $S.2$ and $S.4$ the models with 50 latent topics have the highest log-likelihood. In this case, after each time-step, the distribution over words, for a single sampled topic in $S.2$ and for all topics in $S.4$, is updated using a whole count. This puts more emphasis to counts coming from the held-out data comparing to update scheme $S.3$, and allows the model to adapt to the unseen music piece. Note that the highest log-likelihood is achieved when we use 50 topics and update scheme $S.4$, which suggests that information coming from the held-out data is very useful for prediction.

Table 5.1: Average next-step prediction log-likelihood under Latent Dirichlet Allocation. Results from 10-fold cross-validation using different update schemes for the word distributions. Bold: Best configuration of the hyperparameters α and β for each update scheme and number of latent topics. See text for details.

		$K = 5$		$K = 10$		$K = 50$	
		$S.1$	$S.2$	$S.1$	$S.2$	$S.1$	$S.2$
$\alpha = 0.01$	$\beta = 0.01$	-2.1207	-2.1023	-2.0857	-2.1196	-2.1010	-2.0839
	$\beta = 1$	-2.1233	-2.0970	-2.0798	-2.1221	-2.0956	-2.0781
	$\beta = 5$	-2.1201	-2.0960	-2.0866	-2.1188	-2.0947	-2.0848
$\alpha = 1$	$\beta = 0.01$	-2.0642	-2.0573	-2.0412	-2.0634	-2.0565	-2.0406
	$\beta = 1$	-2.0658	-2.0565	-2.0441	-2.0651	-2.0558	-2.0433
	$\beta = 5$	-2.0664	-2.0607	-2.0535	-2.0657	-2.0599	-2.0527
$\alpha = 5$	$\beta = 0.01$	-2.0619	-2.0449	**-2.0300**	-2.0613	-2.0443	**-2.0296**
	$\beta = 1$	**-2.0603**	**-2.0427**	-2.0310	**-2.0596**	**-2.0421**	-2.0306
	$\beta = 5$	-2.0693	-2.0452	-2.0402	-2.0686	-2.0446	-2.0398
$\alpha = 10$	$\beta = 5$	-2.0732	-2.0478	-2.0388	-2.0726	-2.0473	-2.0384
	$\beta = 10$	-2.0743	-2.0481	-2.0464	-2.0736	-2.0476	-2.0460
	$\beta = 50$	-2.0774	-2.0589	-2.1285	-2.0767	-2.0584	-2.1279
$\alpha = 50$	$\beta = 5$	-2.1255	-2.0925	-2.0655	-2.1249	-2.0919	-2.0652
	$\beta = 10$	-2.1263	-2.0947	-2.0725	-2.1256	-2.0942	-2.0722
	$\beta = 50$	-2.1278	-2.1017	-2.1143	-2.1271	-2.1012	-2.1140
		$S.3$	$S.4$	$S.3$	$S.4$	$S.3$	$S.4$
$\alpha = 0.01$	$\beta = 0.01$	-2.1196	-2.1010	-2.0839	-2.1180	-2.0978	-2.0712
	$\beta = 1$	-2.1221	-2.0956	-2.0782	-2.1205	-2.0923	-2.0659
	$\beta = 5$	-2.1188	-2.0947	-2.0849	-2.1172	-2.0917	-2.0703
$\alpha = 1$	$\beta = 0.01$	-2.0635	-2.0565	-2.0406	-2.0622	-2.0540	-2.0328
	$\beta = 1$	-2.0651	-2.0558	-2.0433	-2.0638	-2.0533	-2.0352
	$\beta = 5$	-2.0657	-2.0600	-2.0527	-2.0644	-2.0573	-2.0430
$\alpha = 5$	$\beta = 0.01$	-2.0613	-2.0443	**-2.0296**	-2.0600	-2.0419	**-2.0238**
	$\beta = 1$	**-2.0596**	**-2.0421**	-2.0306	**-2.0584**	**-2.0399**	-2.0246
	$\beta = 5$	-2.0686	-2.0447	-2.0398	-2.0672	-2.0424	-2.0328
$\alpha = 10$	$\beta = 5$	-2.0726	-2.0473	-2.0384	-2.0713	-2.0451	-2.0317
	$\beta = 10$	-2.0736	-2.0476	-2.0460	-2.0723	-2.0454	-2.0389
	$\beta = 50$	-2.0768	-2.0584	-2.1278	-2.0754	-2.0562	-2.1075
$\alpha = 50$	$\beta = 5$	-2.1249	-2.0920	-2.0652	-2.1230	-2.0890	-2.0581
	$\beta = 10$	-2.1256	-2.0942	-2.0722	-2.1237	-2.0912	-2.0643
	$\beta = 50$	-2.1271	-2.1012	-2.1140	-2.1253	-2.0983	-2.1026

Table 5.2: Average next-step prediction log-likelihood under the Bigram Topic model ($L = 1$). Results from 10-fold cross-validation using different update schemes for the word distributions. Bold: Best configuration of the hyperparameters α and β for each update scheme and number of latent topics. See text for details.

		$K = 5$		$K = 10$		$K = 50$	
		$S.1$	$S.2$	$S.1$	$S.2$	$S.1$	$S.2$
$\alpha = 0.01$	$\beta = 0.01$	-1.7983	-1.7887	-1.7837	-1.7739	-1.7410	-1.7333
	$\beta = 1$	-1.7922	-1.7843	-1.7757	-1.7672	-1.7413	-1.7316
	$\beta = 5$	-1.7910	-1.7831	-1.7839	-1.7765	-1.7621	-1.7531
$\alpha = 1$	$\beta = 0.01$	-1.7817	-1.7726	**-1.7461**	**-1.7379**	**-1.7026**	**-1.6972**
	$\beta = 1$	**-1.7684**	**-1.7611**	-1.7461	-1.7390	-1.7035	-1.6978
	$\beta = 5$	-1.7703	-1.7636	-1.7540	-1.7477	-1.7441	-1.7386
$\alpha = 5$	$\beta = 0.01$	-1.7863	-1.7775	-1.7570	-1.7498	-1.7051	-1.7007
	$\beta = 1$	-1.7777	-1.7709	-1.7482	-1.7423	-1.7035	-1.6992
	$\beta = 5$	-1.7758	-1.7695	-1.7578	-1.7523	-1.7527	-1.7481
$\alpha = 10$	$\beta = 5$	-1.7910	-1.7848	-1.7670	-1.7616	-1.7607	-1.7572
	$\beta = 10$	-1.7922	-1.7865	-1.7815	-1.7764	-1.7972	-1.7930
	$\beta = 50$	-1.8121	-1.8074	-1.8315	-1.8274	-1.9752	-1.9720
$\alpha = 50$	$\beta = 5$	-1.8280	-1.8225	-1.8216	-1.8168	-1.8096	-1.8068
	$\beta = 10$	-1.8326	-1.8275	-1.8282	-1.8238	-1.8384	-1.8357
	$\beta = 50$	-1.8510	-1.8469	-1.8679	-1.8644	-1.9660	-1.9639
		$S.3$	$S.4$	$S.3$	$S.4$	$S.3$	$S.4$
$\alpha = 0.01$	$\beta = 0.01$	-1.7884	-1.7739	-1.7734	-1.7496	-1.7304	-1.6703
	$\beta = 1$	-1.7841	-1.7719	-1.7668	-1.7445	-1.7311	-1.6682
	$\beta = 5$	-1.7833	-1.7715	-1.7764	-1.7547	-1.7525	-1.6887
$\alpha = 1$	$\beta = 0.01$	-1.7724	-1.7574	**-1.7377**	**-1.7154**	**-1.6950**	**-1.6395**
	$\beta = 1$	**-1.7611**	**-1.7488**	-1.7390	-1.7171	-1.6973	-1.6396
	$\beta = 5$	-1.7636	-1.7513	-1.7477	-1.7266	-1.7381	-1.6747
$\alpha = 5$	$\beta = 0.01$	-1.7774	-1.7625	-1.7492	-1.7260	-1.6986	-1.6421
	$\beta = 1$	-1.7709	-1.7576	-1.7421	-1.7197	-1.6990	-1.6420
	$\beta = 5$	-1.7696	-1.7571	-1.7522	-1.7308	-1.7479	-1.6825
$\alpha = 10$	$\beta = 5$	-1.7848	-1.7720	-1.7616	-1.7397	-1.7569	-1.6914
	$\beta = 10$	-1.7865	-1.7739	-1.7763	-1.7540	-1.7930	-1.7229
	$\beta = 50$	-1.8074	-1.7969	-1.8274	-1.8065	-1.9719	-1.8852
$\alpha = 50$	$\beta = 5$	-1.8225	-1.8090	-1.8168	-1.7919	-1.8065	-1.7335
	$\beta = 10$	-1.8274	-1.8140	-1.8238	-1.7994	-1.8355	-1.7605
	$\beta = 50$	-1.8467	-1.8350	-1.8644	-1.8421	-1.9638	-1.8905

Table 5.3: Average next-step prediction log-likelihood under the Variable-gram Topic model with a shallow tree ($\epsilon_1 = 10^{-3}$). Results from 10-fold cross-validation using different update schemes for the word distributions. Bold: Best configuration of the hyperparameters α and β for each update scheme and number of latent topics. See text for details.

		$K = 5$		$K = 10$		$K = 50$	
		$S.1$	$S.2$	$S.1$	$S.2$	$S.1$	$S.2$
$\alpha = 0.01$	$\beta = 0.01$	-2.0348	-1.5914	-1.9699	-1.6662	-1.6575	-1.5788
	$\beta = 1$	-1.7494	-1.3931	-1.6954	-1.4547	-1.5676	-1.4785
	$\beta = 5$	-1.6444	**-1.3113**	-1.6074	**-1.3734**	-1.5729	**-1.4599**
$\alpha = 1$	$\beta = 0.01$	-2.0345	-1.5935	-1.9613	-1.6602	-1.6373	-1.5632
	$\beta = 1$	-1.7472	-1.3973	-1.6842	-1.4614	**-1.5448**	-1.4747
	$\beta = 5$	-1.6381	-1.3206	**-1.5946**	-1.3776	-1.5556	-1.4646
$\alpha = 5$	$\beta = 0.01$	-2.0253	-1.5857	-1.9536	-1.6600	-1.6475	-1.5793
	$\beta = 1$	-1.7435	-1.3969	-1.6787	-1.4642	-1.5569	-1.4990
	$\beta = 5$	-1.6426	-1.3379	-1.5986	-1.4031	-1.5753	-1.5041
$\alpha = 10$	$\beta = 5$	-1.6499	-1.3545	-1.6044	-1.4185	-1.5924	-1.5273
	$\beta = 10$	-1.6194	-1.3632	-1.5972	-1.4292	-1.6333	-1.5736
	$\beta = 50$	**-1.6170**	-1.4805	-1.6534	-1.5642	-1.8434	-1.8011
$\alpha = 50$	$\beta = 5$	-1.6690	-1.3838	-1.6414	-1.4705	-1.6612	-1.6044
	$\beta = 10$	-1.6464	-1.4023	-1.6278	-1.4769	-1.6973	-1.6438
	$\beta = 50$	-1.6495	-1.5220	-1.6857	-1.6043	-1.8807	-1.8499
		$S.3$	$S.4$	$S.3$	$S.4$	$S.3$	$S.4$
$\alpha = 0.01$	$\beta = 0.01$	-1.5885	-1.5891	-1.5697	-1.5671	-1.4459	-1.4676
	$\beta = 1$	-1.0133	-1.0103	-0.9974	-0.9899	-0.9553	-0.9459
	$\beta = 5$	-0.9973	-0.9724	-0.9915	-0.9546	-0.9963	-0.9320
$\alpha = 1$	$\beta = 0.01$	-1.5858	-1.5861	-1.5614	-1.5590	-1.4285	-1.4490
	$\beta = 1$	-1.0097	-1.0064	-0.9893	-0.9814	**-0.9454**	-0.9335
	$\beta = 5$	**-0.9931**	**-0.9671**	-0.9843	**-0.9460**	-0.9900	**-0.9240**
$\alpha = 5$	$\beta = 0.01$	-1.5771	-1.5773	-1.5530	-1.5502	-1.4265	-1.4416
	$\beta = 1$	-1.0047	-1.0014	**-0.9814**	-0.9732	-0.9488	-0.9316
	$\beta = 5$	-0.9971	-0.9697	-0.9896	-0.9487	-1.0048	-0.9314
$\alpha = 10$	$\beta = 5$	-1.0007	-0.9721	-0.9918	-0.9499	-1.0158	-0.9375
	$\beta = 10$	-1.0793	-1.0318	-1.0842	-1.0175	-1.1430	-1.0255
	$\beta = 50$	-1.3506	-1.2882	-1.3998	-1.3108	-1.5882	-1.4234
$\alpha = 50$	$\beta = 5$	-1.0074	-0.9772	-1.0061	-0.9604	-1.0424	-0.9557
	$\beta = 10$	-1.0923	-1.0415	-1.1004	-1.0288	-1.1691	-1.0430
	$\beta = 50$	-1.3758	-1.3054	-1.4215	-1.3254	-1.6037	-1.4320

Table 5.4: Average next-step prediction log-likelihood under the Variable-gram Topic model with a deep tree ($\epsilon_1 = 10^{-4}$). Results from 10-fold cross-validation using different update schemes for the word distributions. Bold: Best configuration of the hyperparameters α and β for each update scheme and number of latent topics. See text for details.

		$K = 5$		$K = 10$		$K = 50$	
		$S.1$	$S.2$	$S.1$	$S.2$	$S.1$	$S.2$
$\alpha = 0.01$	$\beta = 0.01$	-2.9424	-2.0673	-2.2419	-1.8232	-1.6541	-1.5709
	$\beta = 1$	-1.9065	-1.4795	-1.7202	-1.4724	-1.5465	-1.4559
	$\beta = 5$	-1.6583	-1.3101	-1.5793	**-1.3489**	-1.5428	**-1.4187**
$\alpha = 1$	$\beta = 0.01$	-2.9484	-2.0658	-2.2287	-1.8149	-1.6260	-1.5556
	$\beta = 1$	-1.9126	-1.4808	-1.7105	-1.4708	**-1.5288**	-1.4594
	$\beta = 5$	-1.6576	-1.3161	-1.5786	-1.3574	-1.5440	-1.4447
$\alpha = 5$	$\beta = 0.01$	-2.9330	-2.0537	-2.2187	-1.8059	-1.6247	-1.5595
	$\beta = 1$	-1.9059	-1.4812	-1.7135	-1.4799	-1.5458	-1.4870
	$\beta = 5$	-1.6504	-1.3207	-1.5837	-1.3758	-1.5658	-1.4829
$\alpha = 10$	$\beta = 5$	-1.6607	-1.3339	-1.5862	-1.3838	-1.5822	-1.5090
	$\beta = 10$	-1.6048	**-1.3045**	**-1.5772**	-1.3844	-1.6304	-1.5524
	$\beta = 50$	**-1.6016**	-1.4051	-1.6407	-1.5081	-1.8243	-1.7610
$\alpha = 50$	$\beta = 5$	-1.6724	-1.3519	-1.6203	-1.4264	-1.6509	-1.5870
	$\beta = 10$	-1.6199	-1.3350	-1.6085	-1.4238	-1.6897	-1.6216
	$\beta = 50$	-1.6247	-1.4353	-1.6692	-1.5444	-1.8791	-1.8210
		$S.3$	$S.4$	$S.3$	$S.4$	$S.3$	$S.4$
$\alpha = 0.01$	$\beta = 0.01$	-2.4513	-2.4635	-2.1261	-2.1567	-1.8355	-1.9320
	$\beta = 1$	-1.1746	-1.1643	-1.0913	-1.0777	-1.0221	-1.0087
	$\beta = 5$	-0.9856	-0.9616	-0.9573	-0.9235	**-0.9540**	-0.8997
$\alpha = 1$	$\beta = 0.01$	-2.4515	-2.4621	-2.1156	-2.1493	-1.8145	-1.9091
	$\beta = 1$	-1.1732	-1.1626	-1.0855	-1.0722	-1.0095	-0.9949
	$\beta = 5$	-0.9852	-0.9603	**-0.9551**	-0.9201	-0.9571	**-0.8984**
$\alpha = 5$	$\beta = 0.01$	-2.4405	-2.4530	-2.1023	-2.1324	-1.8103	-1.8973
	$\beta = 1$	-1.1680	-1.1569	-1.0807	-1.0659	-1.0200	-0.9975
	$\beta = 5$	**-0.9816**	-0.9565	-0.9572	**-0.9192**	-0.9727	-0.9068
$\alpha = 10$	$\beta = 5$	-0.9866	-0.9597	-0.9599	-0.9210	-0.9836	-0.9121
	$\beta = 10$	-0.9891	**-0.9534**	-0.9873	-0.9350	-1.0447	-0.9508
	$\beta = 50$	-1.1957	-1.1411	-1.2406	-1.1650	-1.4001	-1.2655
$\alpha = 50$	$\beta = 5$	-0.9909	-0.9629	-0.9738	-0.9308	-1.0107	-0.9284
	$\beta = 10$	-0.9958	-0.9578	-1.0013	-0.9451	-1.0701	-0.9674
	$\beta = 50$	-1.2099	-1.1515	-1.2555	-1.1757	-1.4225	-1.2811

The performance of the Bigram Topic model is shown in Table 5.2. The best setting for the concentration parameters of the prior distributions are given by $\alpha = 1$ and $\beta = \{0.01, 1\}$. It is interesting to note that when we have a small number of topics, $K = 5$, the larger value of β is preferred. But when we increase the number of latent topics, allowing the model more flexibility to capture different bigram statistics, the model prefers sharper posterior distributions over words.

In the topic Bigram, performance increases as we increase the number of topics, irrespective of the update scheme that we use. This suggests that, although 50 topics were leading to overfitting in Latent Dirichlet Allocation, when we consider bigram statistics the different topics are used more effectively, capturing structure that was not possible to model by considering unigram statistics. In fact, we can see that there is a bigger improvement in performance when we increase the number of topics from 10 to 50, than from 5 to 10.

With respect to the update scheme for the distributions over words, we can see that there is a marginal improvement when we use $S.2$ or $S.3$ compared to $S.1$, with the former two being almost the same, and we have a further improvement when we use update $S.4$. This again suggests that considering information coming from the unseen piece during prediction is beneficial.

The prediction log-likelihood under the Variable-gram Topic models is shown in Tables 5.3 and 5.4, for the model using the shallow and the deep Dirichlet-VMM tree, respectively. Looking at update scheme $S.1$, we can see that the best value for the concentration parameter over topic distributions, α, depends on the number of topics that we use. For $K = 5$, the value $\alpha = 10$ is optimal for both models (shallow and deep tree). For $K = 10$, the value $\alpha = 10$ is optimal for the model with the shallow tree, but for the model with the deep tree a smaller value, $\alpha = 1$ is preferred. The smaller value is also preferred when we use 50 latent topics. A small α value corresponds to sharper posterior distributions, i.e. a smaller number of topics is used for each document. When we have many topics available and consider longer contextual dependencies, different topics are able to capture the finer structures in melody and each music piece wants to be explained by only a few "specialised" topics. Models using the shallower tree and fewer topics do not have as much representational power and thus we need to use many of the available topics in each piece.

When using update schemes $S.1$, $S.3$ and $S.4$, performance increases as we increase the number of topics. For update scheme $S.2$ we have the opposite behaviour, with performance decreasing as we increase the number of topics. In update scheme $S.2$ after each time-step we sample a topic and update only the word distributions of that topic. This means that this information is not available for future time-steps that are allocated to different topics. The more topics we have, the more the information from the new piece is likely to be dispersed. Note that for all the models, using update scheme $S.2$ gives higher log-likelihood than using update scheme $S.1$, which is further increased with update scheme $S.3$ and is maximised with update scheme $S.4$. This signifies the importance of the information coming from the unseen piece in the prediction task. Although the latent topics increase the representational power of the model and lead to better performance in the prediction task, the information coming from the unseen piece is still the most predictive feature of the future.

5.3.1 Qualitative evaluation

An attractive property of topic models when applied in natural language tasks is that they discover meaningful topics. In this Section we examine aspects of the latent topic allocations and the inferred parameters of the Variable-gram Topic model and analyse them with respect to musical features.

Figure 5.2 shows scatter plots of the number of times each topic has been allocated in pieces from the key of G (x-axis) and in pieces from the key of D (y-axis) for different models. These are the topic allocations for the training data after the Gibbs sampler has converged. We can see that in many of the cases the models have learned to distinguish the key, as the per key topic allocations are negatively correlated. In all models we can identify some topics that are shared across the two keys, but we also find topics that are "key specific", that is topics that tend to be allocated in pieces from a single key. The latter happens more often in models with many latent topics (3rd column), in which case models have enough representational resources to dedicate some of them for capturing key specific statistics.

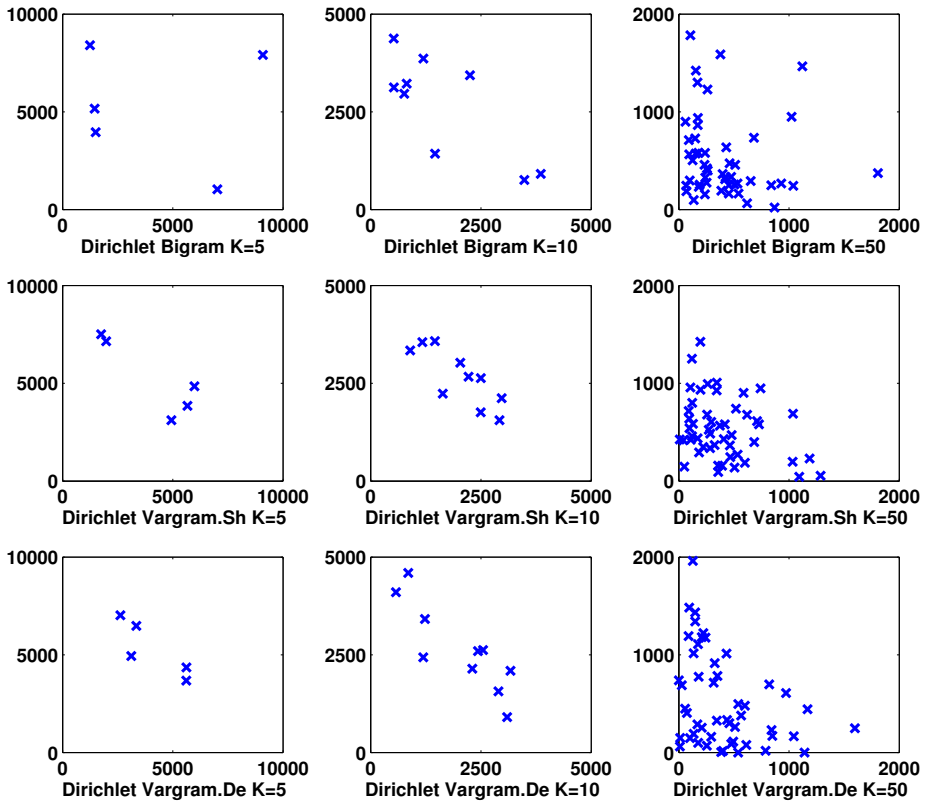


Figure 5.2: Scatter plots of topic allocations with respect to the key. x-axis: number of times each latent topic is allocated in pieces from the key of G, y-axis: number of times each latent topic is allocated in pieces from the key of D. Topic allocations are taken from the training data after Gibbs sampling has converged. The per key topic allocations are negatively correlated. In most models we can identify some topics that are key-specific. These occur more often in models with many latent topics (3rd column). See text for more details.

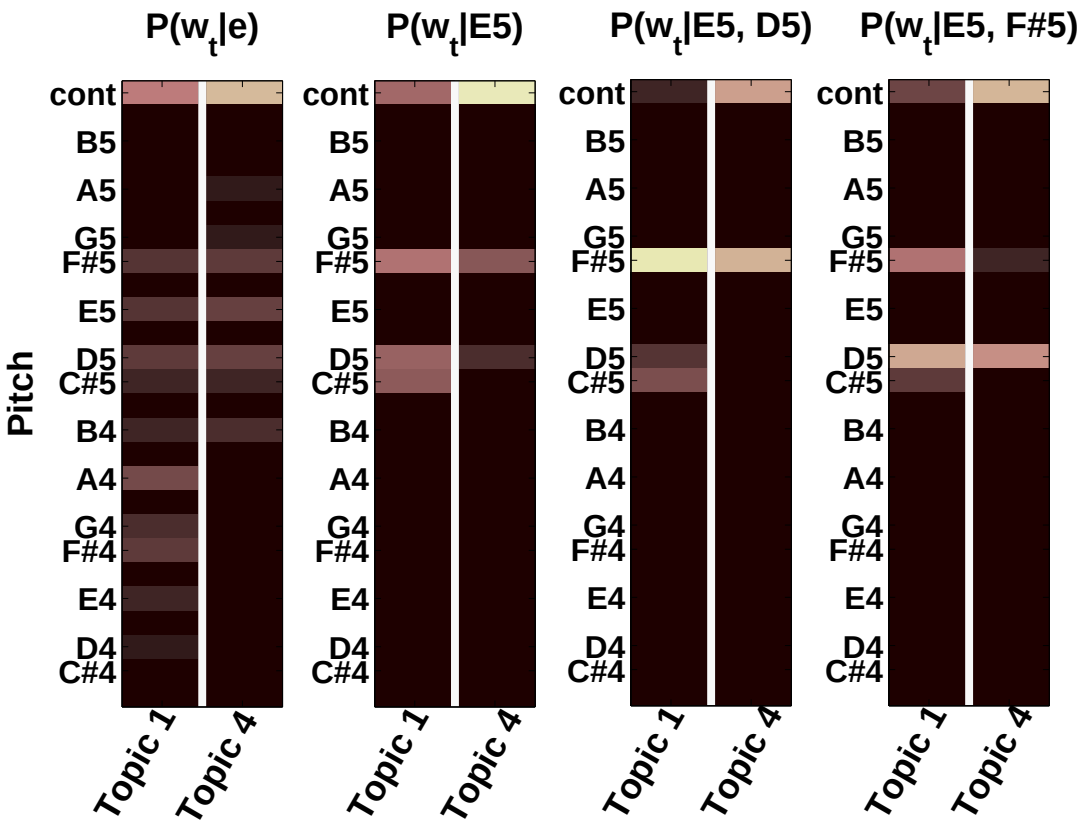


Figure 5.3: Conditional distributions over notes for 2 topics that are mostly assigned in pieces from the key of D. Each distribution is conditioned on a different context. Model: Variable-gram.$De.K_{50}$. The distributions are successively sharper. Topic 4 models slower parts of the melody (prefers "continuation"). See text for more details.

Figure 5.3 shows the conditional distributions over notes for 2 topics that are used in pieces from the key of D for 4 different contexts. The first context is the empty string, i.e. the distribution of the root node of the Dirichlet-VMM tree corresponding to each of the topics. The second context is note $E5$ and the other two are note $E5$ preceded by $D5$ and $F^{\#}5$ respectively. We can see that the conditional distributions are successively sharper. Topic 4 tends to prefer "continuation", thus modelling slower parts of the melody, whereas topic 1 assigns lower probability to "continuation", especially given the longer contexts. At the same time we can observe that, depending on the context, the topics prefer different notes. For instance, if there is an upward movement in the context, $D5$ followed by $E5$ (subfigure (c)), Topic 4 wants to continue going upwards, i.e. high probability on $F^{\#}5$, and vice versa for a downward movement (subfigure (d)).

5.4 Comparative Analysis

In this Section we evaluate the Variable-gram Topic model by comparing its performance with Latent Dirichlet Allocation, the Bigram Topic model and the Dirichlet-VMM models examined in Chapter 3. Model performance is assessed using the average next-step prediction log-likelihood of a held-out testing set comprising 64 music pieces and the Maximum Mean Discrepancy between the testing pieces and model samples.

We select the model hyperparameters by looking at the results from the cross-validation experiment discussed above. More specifically, for each model type we select the α and β values that maximise the average log-likelihood of the validation data using update scheme $S.3$. The selected values for each model are shown in Table 5.5.

Table 5.5: Optimal hyperparameter settings for the topic models used in the comparative analysis.

	$K=5$		$K=10$		$K=50$	
	α	β	α	β	α	β
LDA	5	1	5	1	5	1
Topic Bigram	1	1	1	0.01	1	0.01
Topic Variable-gram.Sh	1	5	5	1	1	1
Topic Variable-gram.De	5	5	1	5	0.01	5

We report results that are averaged across different folds. More specifically, for each model type we use the 10 trained models from the cross-validation experiment above to compute the evaluation metrics on the testing data and we report the mean and the standard deviation of the results across folds.

5.4.1 Next-Step Prediction

Table 5.6 shows the average next-step prediction log-likelihood of the testing data under different models. The first thing we can observe is that introducing latent topics is useful for modelling melody, as all the topic models perform significantly better than their non-topic counterparts. More specifically, if we do not consider any temporal dependencies then LDA is better than the empirical marginal

distribution. Similarly, the Bigram Topic model is better than the Dirichlet-Bigram and the Variable-gram Topic models are better than the Dirichlet-VMMs.

Table 5.6: Average next step prediction log-likelihood of the testing data under different models. For each model type, we use the 10 trained models from cross-validation and compute the average next-step prediction log-likelihood of the testing data (64 pieces) under each of these models. The mean and standard deviation are computed across the results from the 10 trained models. Including latent topics improves performance, but does not overcome the need for a systematic model of contextual information. Incorporating information from the testing piece (Global-Local combinations and update $S.3$) increases the predictive ability of the models. See text for more details.

Stationary Models	Mean ± St.Dev.		
Empirical Marginal	-2.2217 ± 0.0009		
Dirichlet-Bigram	-1.9171 ± 0.0019		
Dirichlet-VMM.Sh	-1.6084 ± 0.0030		
Dirichlet-VMM.De	-1.5663 ± 0.0040		
Global-Local Combinations			
GL_1-fixed ($.De$)	-1.4454 ± 0.0038		
GL_1-dynamic ($.Sh$)	-1.0393 ± 0.0054		
GL_2 ($.Sh$)	-1.4309 ± 0.0052		
GL_3-fixed ($.De$)	-1.3072 ± 0.0021		
GL_3-dynamic ($.De$)	-1.0905 ± 0.0012		
Topic Models – Update $S.1$	$K = 5$	$K = 10$	$K = 50$
LDA	-2.0494 ± 0.0057	-2.0329 ± 0.0037	-2.0217 ± 0.0030
Topic Bigram	-1.7628 ± 0.0066	-1.7531 ± 0.0076	-1.7077 ± 0.0039
Topic Variable-gram.Sh	-1.6183 ± 0.0082	-1.6601 ± 0.0091	-1.5311 ± 0.0055
Topic Variable-gram.De	-1.6248 ± 0.0088	-1.5500 ± 0.0045	-1.5223 ± 0.0076
Topic Models – Update $S.3$	$K = 5$	$K = 10$	$K = 50$
LDA	-2.0490 ± 0.0056	-2.0325 ± 0.0037	-2.0214 ± 0.0030
Topic Bigram	-1.7543 ± 0.0062	-1.7405 ± 0.0072	-1.6971 ± 0.0036
Topic Variable-gram.Sh	-1.0027 ± 0.0076	-0.9889 ± 0.0049	-0.9518 ± 0.0032
Topic Variable-gram.De	-0.9879 ± 0.0042	-0.9589 ± 0.0019	-0.9658 ± 0.0040

Furthermore, we can see that in most cases log-likelihood increases as we increase the number of topics. This improvement is only marginal in the case of LDA, but for models that consider temporal dependencies, the number of topics has a considerable effect on performance. One exception is the topic Variable-gram.Sh, where for update scheme $S.1$ performance drops when we increase the number

of topics from 5 to 10. This is due to our model selection, which is based on optimising performance using update scheme $S.3$ rather than $S.1$. Finally, we also have a marginal drop in performance in the topic Variable-gram.De using update scheme $S.3$ when we increase the topics from 10 to 50.

A second observation is that modelling temporal dependencies is very important in melody. The Dirichlet-VMM, which is able to capture both large and small order Markov dependencies performs better than the Bigram Topic model and LDA, which model only first order and no temporal dependencies, respectively. Therefore, the predictive information of the context is higher than that of the latent topics. This is also evident by the fact that performance improves as we consider longer contexts, with the Bigram Topic model being significantly better than LDA and with the Variable-gram Topic models being consistently better than all other models, with the model using the deeper tree being best. This is true for both update schemes, with the exception of models with 50 latent topics, where the Variable-gram.Sh is marginally better than the Variable-gram.De, when we use update scheme $S.3$.

Finally, it is interesting to compare model performance using update schemes $S.1$ and $S.3$. We can see that performance of the Variable-gram Topic models improves significantly when the distributions over notes are updated during prediction, whereas the improvement in LDA and the Bigram Topic model is only marginal. This signifies that longer contexts can have different meaning across music pieces, and thus incorporating information coming from the testing piece can lead to much better prediction.

5.4.2 Maximum Mean Discrepancy of String Kernels

In this Section we evaluate the performance of topic models by looking at the squared MMD between data sequences and model samples. Similar to Chapters 3 and 4, we generate 10 sets of samples from each topic model, with each set sampled from a model trained on a different data fold. We then compute the Maximum Mean Discrepancy between: a) each set of samples and the set of testing sequences and b) each set of samples and a set of randomly selected training sequences from the corresponding fold. Note that all sets contain 64 sequences.

For the topic models we generate samples in two ways. In the first case, we

sample both the latent topics and the corresponding notes (words). To do this we initialise the distribution over topics to the prior Dirichlet distribution and proceed sequentially. At every time-step we sample a latent topic from our current estimate of the posterior distribution over topics, sample a note given the topic and subsequently update the posterior distribution over topics by adding the sampled topic to the counts. We refer to this sampling scheme as "sampling from the prior". In the second approach, which is denoted with the * symbol in the figures, we first run Gibbs sampling on the testing sequences to get a set of topic allocations and then we perform sampling from the model given the topic allocations. This restricts the sampling noise, but is not directly comparable with the non-topic models, as it uses information from the test pieces captured by the latent topic allocations. We refer to this sampling scheme as "sampling given topics".

As in the previous Section, we assess the performance of topic models by comparing to the empirical marginal distribution and the Dirichlet-VMM models from Chapter 3. Additionally, we report the squared MMD using: a) samples drawn from a Uniform distribution, which has no information regarding the structure of the data and b) testing sequences and training sequences, which are drawn from the same theoretical distribution. These act as indicators to what we can expect in terms of the worst and best performance, respectively.

Table 5.7 shows the mean and standard deviation of the estimated squared MMD for different models, computed using the testing sequences and the 10 sets of model samples. For Latent Dirichlet Allocation, using the first sampling approach is problematic and is not reported in the Table. As described in Section 2.3.1.1, we use a "continuation" value to represent longer notes, so that any note longer than an eighth is encoded by the corresponding pitch in the first time-step followed by "continuation" for the remaining time-steps. This results in "continuation" occurring roughly 30% of the time. In LDA many of the topics give high probability to "continuation" and when we sample from the prior it is often the case that the resulting sequence is dominated by "continuation". In the Variable-gram Topic model this is not a problem, as the probability of getting continuation repeatedly decreases due to the dependency on the context.

Table 5.7: Mean, standard deviation, minimum and maximum of the estimated squared MMD between testing sequences and samples generated from different models. Results for the $(4,1)$ and the normalised $(4,1)$ mismatch kernels. Adding latent topics does not improve performance over the Dirichlet-VMM when using the mismatch kernel, but it does with the normalised mismatch kernel. The latter kernel considers both the inter and the intra-sequence structure. See text for more details.

	$(4,1)$ Mismatch Kernel			Normalised $(4,1)$ Mismatch Kernel		
	Mean $\pm$ St.Dev.	Min	Max	Mean $\pm$ St.Dev.	Min	Max
Uniform $\mathcal{U}(\lvert\Sigma\rvert)$	411.69 ± 1.31	409.34	413.13	0.1657 ± 0.0011	0.1639	0.1668
EmpMarg	129.89 ± 10.13	116.04	147.08	0.0644 ± 0.0054	0.0543	0.0723
TrainData	6.22 ± 4.58	1.33	16.71	0.0030 ± 0.0015	0.0002	0.0048
VMM	173.58 ± 8.56	159.81	187.84	0.0269 ± 0.0046	0.0200	0.0360
Dir-Bigram	91.81 ± 6.10	83.67	102.18	0.0373 ± 0.0024	0.0342	0.0407
Dir-Vmm.Sh	9.20 ± 4.97	2.71	19.79	0.0257 ± 0.0046	0.0163	0.0342
Dir-VMM.De	10.04 ± 5.23	2.18	18.71	0.0256 ± 0.0034	0.0205	0.0296
Bigram.K_5	123.59 ± 23.93	88.51	167.86	0.0403 ± 0.0087	0.0311	0.0579
Vargram.$Sh.K_5$	15.43 ± 11.28	0.42	31.54	0.0210 ± 0.0031	0.0170	0.0253
Vargram.$De.K_5$	20.10 ± 11.48	2.67	36.32	0.0243 ± 0.0060	0.0128	0.0329
Bigram.K_{10}	120.25 ± 29.51	80.52	178.15	0.0393 ± 0.0097	0.0264	0.0558
Vargram.$Sh.K_{10}$	18.68 ± 5.09	11.00	29.30	0.0233 ± 0.0044	0.0177	0.0303
Vargram.$De.K_{10}$	16.94 ± 5.89	8.99	25.70	0.0217 ± 0.0030	0.0180	0.0275
Bigram.K_{50}	96.10 ± 18.09	66.33	123.07	0.0271 ± 0.0058	0.0182	0.0370
Vargram.$Sh.K_{50}$	30.63 ± 9.96	12.01	46.73	0.0163 ± 0.0029	0.0117	0.0216
Vargram.$De.K_{50}$	39.00 ± 15.16	25.91	71.41	0.0160 ± 0.0037	0.0106	0.0209
LDA.K^*_5	85.00 ± 5.57	76.41	92.64	0.0397 ± 0.0038	0.0350	0.0472
Bigram.K^*_5	71.64 ± 4.34	66.29	78.78	0.0272 ± 0.0010	0.0258	0.0285
Vargram.$Sh.K^*_5$	12.89 ± 6.35	3.77	23.56	0.0201 ± 0.0046	0.0144	0.0274
Vargram.$De.K^*_5$	12.16 ± 5.78	3.18	18.57	0.0224 ± 0.0056	0.0149	0.0320
LDA.K^*_{10}	80.09 ± 10.56	66.51	100.72	0.0378 ± 0.0039	0.0329	0.0436
Bigram.K^*_{10}	68.74 ± 12.24	47.84	88.27	0.0257 ± 0.0036	0.0201	0.0307
Vargram.$Sh.K^*_{10}$	16.95 ± 8.91	5.13	35.59	0.0228 ± 0.0054	0.0163	0.0340
Vargram.$De.K^*_{10}$	14.24 ± 5.34	4.97	21.71	0.0216 ± 0.0035	0.0153	0.0256
LDA.K^*_{50}	82.56 ± 9.35	70.85	101.67	0.0399 ± 0.0046	0.0338	0.0464
Bigram.K^*_{50}	52.31 ± 4.45	44.29	59.73	0.0170 ± 0.0025	0.0131	0.0204
Vargram.$Sh.K^*_{50}$	9.45 ± 5.21	2.95	20.03	0.0119 ± 0.0032	0.0084	0.0177
Vargram.$De.K^*_{50}$	16.96 ± 9.36	3.40	32.57	0.0123 ± 0.0040	0.0047	0.0194

The results from Table 5.7 are also shown in Figures 5.4 and 5.5 for the mismatch and the normalised mismatch kernels, respectively. The bars for $K = 1$ show the Dirichlet-VMM models, which can be seen as having a single latent topic. In the Figures we also present barplots for the squared MMD computed between model samples and training sequences (bottom).

Looking at Figure 5.4 we can see that the comparative performance of the models under the MMD evaluation is somewhat different to the one in the next-step prediction task. In the case of the mismatch kernel, adding latent topics does not improve performance, especially when we consider samples from the first sampling scheme (prior). The squared MMD for the Variable-gram Topic models is higher than for the corresponding Dirichlet-VMM and tends to increase further as we increase the number of topics. Additionally the model with the deeper tree performs worse than the model with the shallow tree. Note that this is also true for the squared MMD computed using the training data, which suggests that models with high representational power – many latent topics or complex contextual information – are possibly underfitting the data, allowing some of the probability mass to be assigned to configurations that have never occurred. When we constrain the sampling procedure, by clamping the latent topics to the testing data, performance improves, with all the Variable-gram Topic models being closer to the Dirichlet-VMM ones and with the Variable-gram.Sh using 50 topics being the best.

Another important observation is that irrespective of whether we use latent topics or not, when modelling melody capturing the temporal dependencies is very important. Both variable-gram models are consistently and notably better than the equivalent bigrams, using either sampling scheme and the bigram models are always better than the empirical marginal distribution and LDA (Table 5.7).

Using the normalised $(4, 1)$ mismatch kernel, Figure 5.5, gives a comparative performance that is qualitatively similar to the one we get from the next-step prediction task. Namely, using more latent topics and longer contextual information tends to improve performance. This suggests that the ability of the topic models to differentiate between sequences, by allowing a unique distribution over topics in each sampled sequence, leads to improved results when we take into account both the inter-sequence and the intra-sequence structure. We will examine this further in the next Section, where we visualise the mismatch kernel matrices.

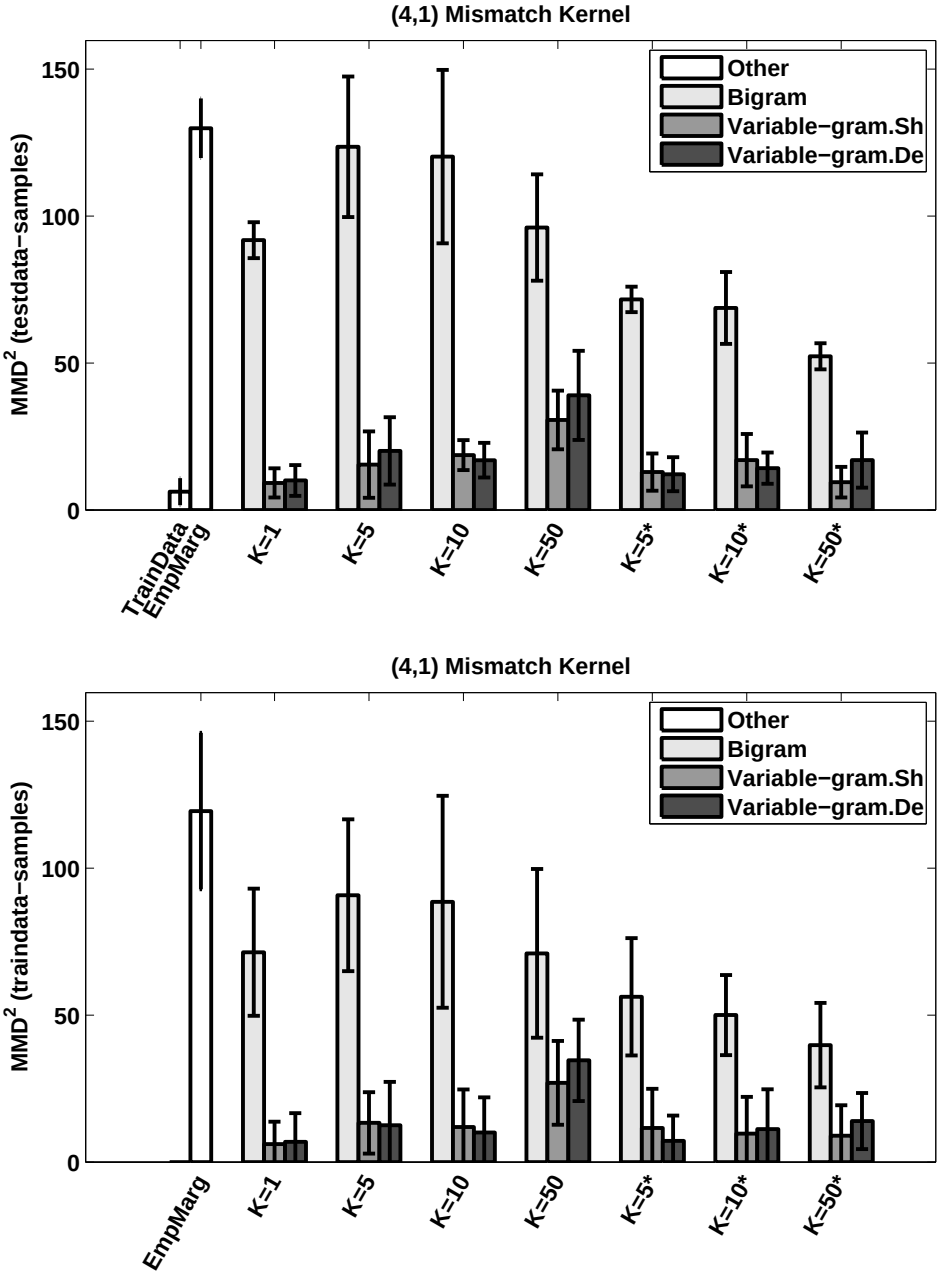


Figure 5.4: Barplots depicting the mean and standard deviation of the estimated squared MMD between data sequences and samples generated from different models. Top: squared MMD between testing sequences and model samples. Bottom: squared MMD between training sequences and model samples. Results using the $(4, 1)$ mismatch kernel. See text for details.

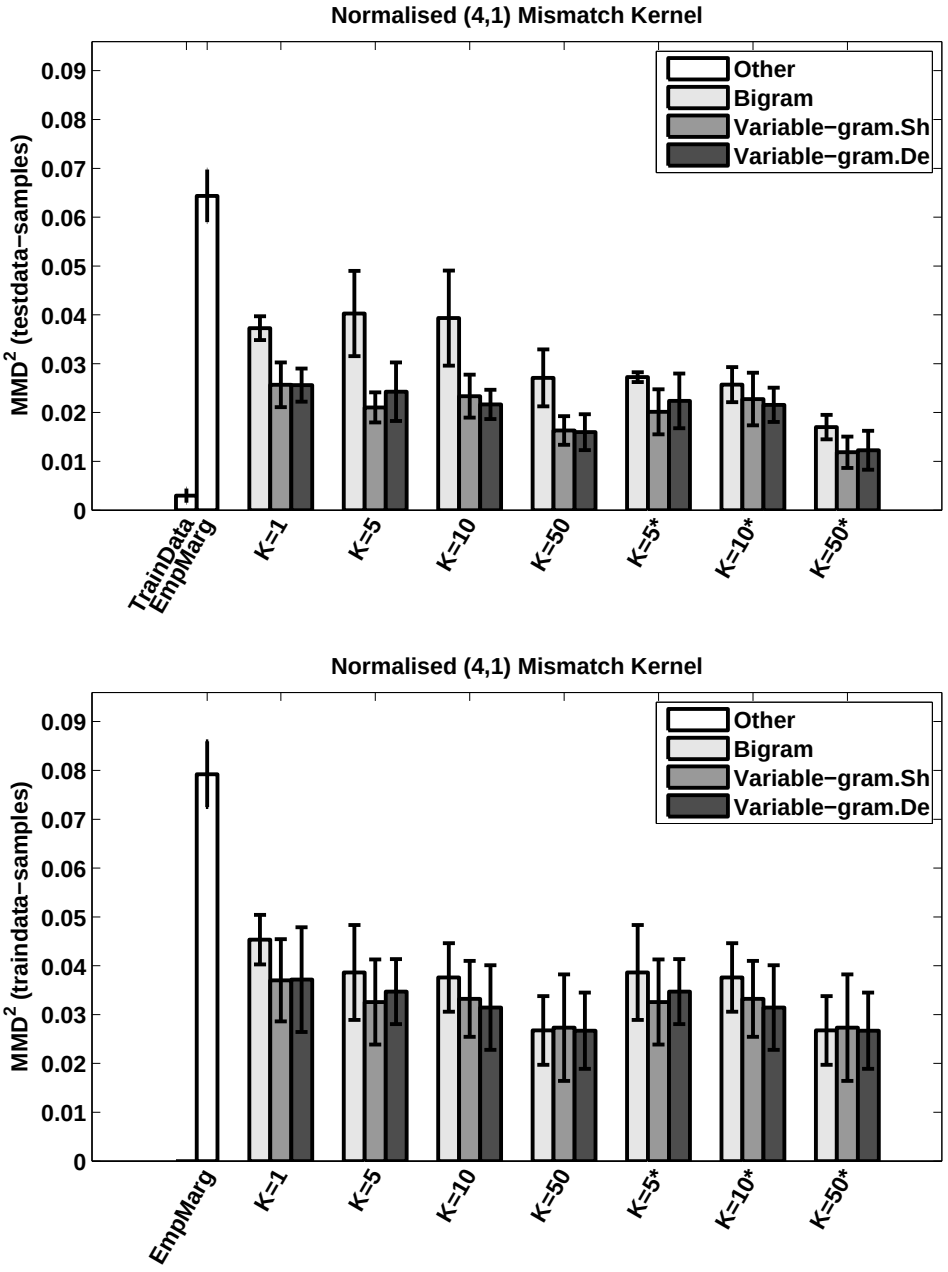


Figure 5.5: Barplots depicting the mean and standard deviation of the estimated squared MMD between data sequences and samples generated from different models. Top: squared MMD between testing sequences and model samples. Bottom: squared MMD between training sequences and model samples. Results using the normalised $(4, 1)$ mismatch kernel. See text for details.

5.4.2.1 Visualising the Kernel Matrices

Figure 5.6 shows the $(4, 1)$ mismatch kernel matrices between testing sequences and samples from different models, where we have truncated very high values to $1,000$, in order to better visualise the structure of the kernels. The kernels for the topic models are computed using samples from the first approach, that is initialising the distribution over topics to the prior and sequentially updating it using update scheme $S.1$. What we can see here, is that in the Variable-gram Topic models as we increase the number of topics the overall level of similarity in the sample-sample part of the matrix decreases, while the level of similarity in the test-sample part remains fairly high. This means that sequences generated from the topic models are less similar to each other when compared to sequences generated from the Dirichlet-VMM, while still being similar to true data sequences albeit somewhat less than the ones from the Dirichlet-VMM. Furthermore, we can observe that in the kernel matrices for the topic models, the elements of the diagonal in the sample-sample part have higher values than the off-diagonal ones, and this difference becomes higher as we increase the number of topics. This indicates that the topic models begin to differentiate between inter and intra-sequence structure, which is not the case with the Dirichlet-VMM.

Looking at the normalised $(4, 1)$ mismatch kernel matrices in Figure 5.7 we can see that the sample-sample part for the Variable-gram Topic models with 50 topics is the one that most resembles the test-test part of the matrix. As already mentioned in previous chapters, the normalised kernel considers both the inter and the intra-sequence structure, as each element of the kernel is divided by the square root of the corresponding diagonal elements (see Equation 2.8). Note that the intra-sequence structure is ignored when using the mismatch kernel as the estimator for the squared MMD excludes the diagonal elements (see Equation 2.7). The ability of the topic models to distinguish between inter and intra-sequence structure is evident in the normalised kernel and this why performance under the MMD using this kernel improves as we increase the number of topics. The inferior performance of the Variable-gram Topic models to the Dirichlet-VMMs when using the non-normalised mismatch kernel can also be understood as a consequence of this property. The Variable-gram Topic model encodes a generative process that needs to generalise over whole sequences. On the contrary, the Dirichlet-VMM captures melodic structure at a motif level. Therefore, we can expect

the sequences generated by the Variable-gram Topic models to be piecewise less similar to true data sequences, when compared to sequences generated by the Dirichlet-VMM, as the number of configurations that must be probable under the former is considerably higher.

Finally, Figure 5.8 shows the truncated $(4, 1)$ mismatch kernel between testing sequences and samples from different topic models obtained using the second sampling scheme, i.e. by clamping the latent topics to a topic assignment for each testing piece and sampling the notes conditioned on the topics. An interesting observation here is that for the topic models using 50 topics (bottom row), the diagonal of the test-sample submatrix tends to have higher values than the rest of the elements in the test-sample part, which suggests that the latent topic is highly informative of the observed note, resulting in samples having more shared occurrences with the test piece whose topic allocations are used.

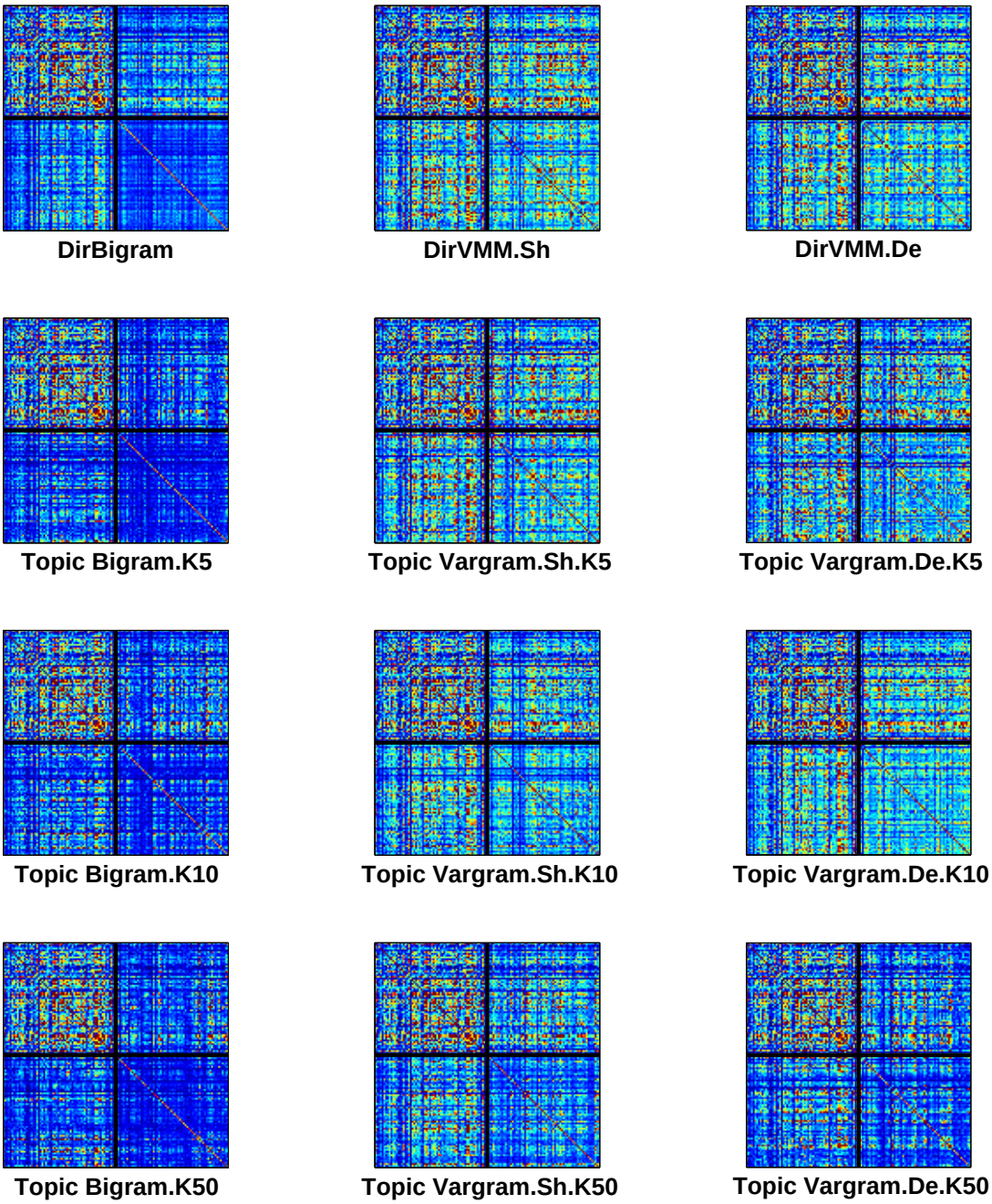


Figure 5.6: The $(4, 1)$ mismatch kernels between testing sequences (rows:1-64) and model samples (rows:65-128) for different models. Colour values range from 0 (dark blue–smallest element across kernels) to $1,000$ (dark red). Values larger than $1,000$ are truncated. Samples from the topic models are drawn by initialising the distribution over topics to the prior and sequentially updating it using update scheme $S.1$. See text for details.

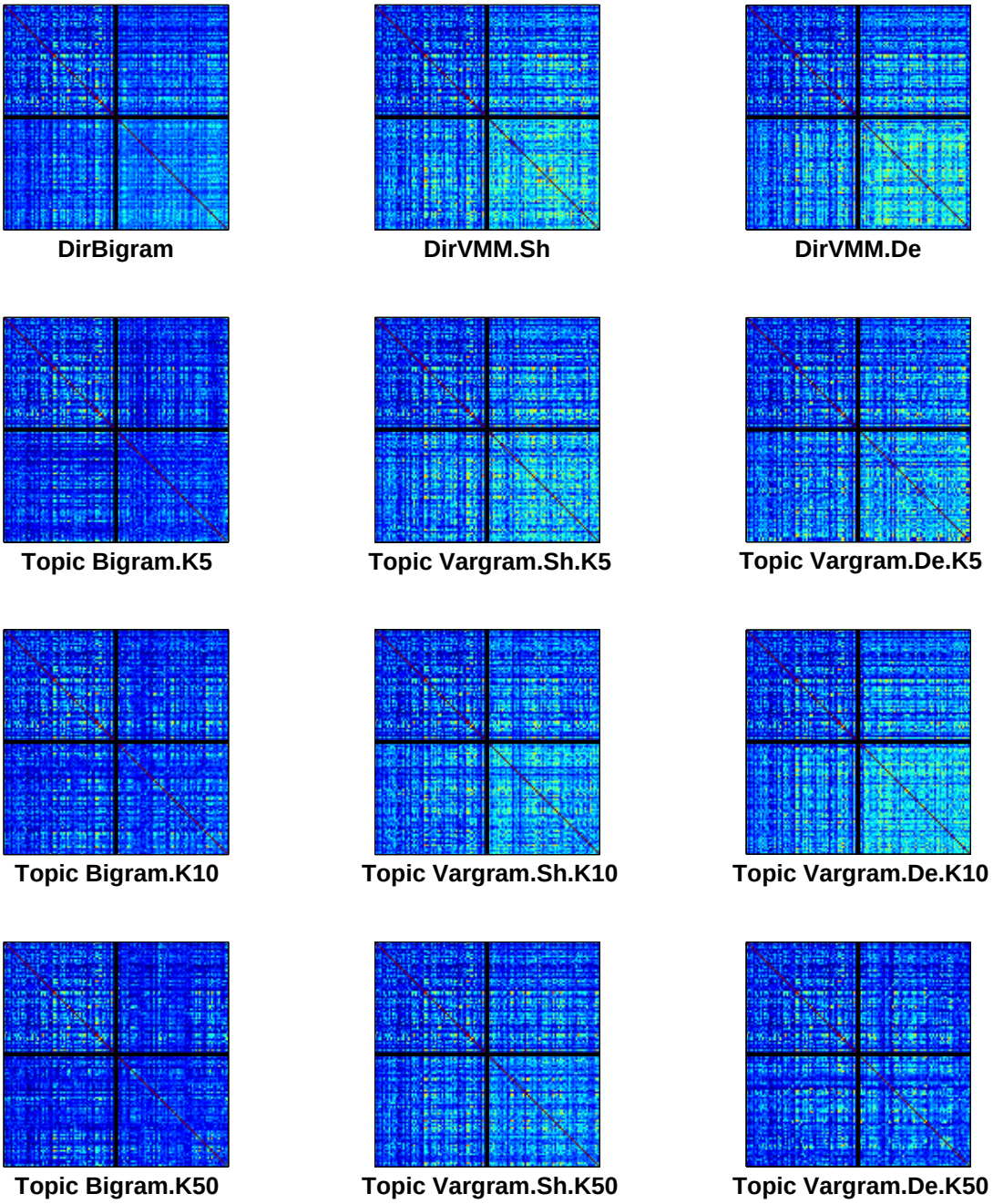


Figure 5.7: The normalised $(4, 1)$ mismatch kernels between testing sequences (rows:1-64) and model samples (rows:65-128) for different models. Samples from the topic models are drawn by initialising the distribution over topics to the prior and sequentially updating it using update scheme $S.3$. See text for details.

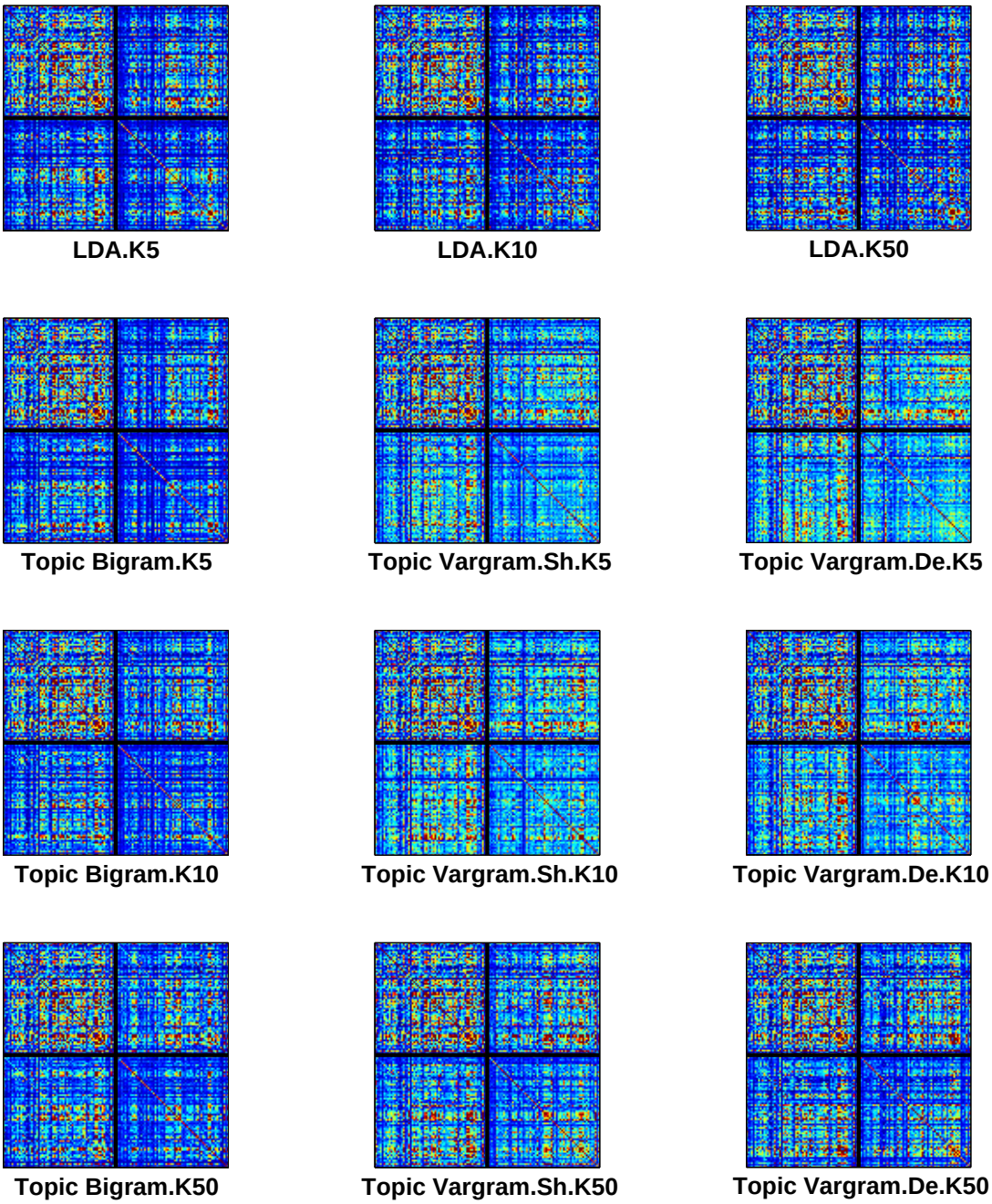


Figure 5.8: The $(4, 1)$ mismatch kernels between testing sequences (rows:1-64) and model samples (rows:65-128) for different models. Colour values range from 0 (dark blue–smallest element across kernels) to $1,000$ (dark red). Values larger than $1,000$ are truncated. Samples from the topic models are drawn by running the Gibbs sampler to get the latent topic assignments for each testing piece and then sampling notes given the allocated topics. See text for details.

5.5 Discussion

In this Chapter we developed the Variable-gram Topic model, which couples the latent topic formalism with an expressive model of contextual information for the problem of modelling melodic sequences. Using the next-step prediction log-likelihood we showed that the model outperforms a number of related methods. Our comparative analysis revealed that although latent topics improve performance when modelling melody, they do not overcome the need for a systematic temporal model. We believe it would be interesting to apply the Variable-gram Topic model in other sequential data domains where the sequences are non-stationary and exhibit temporal dependencies of considerable complexity.

Using the MMD evaluation we found that samples from the Variable-gram Topic models exhibit different inter and intra-sequence structure – a desirable property, which is in accordance to what we observe in true musical pieces. However, this flexibility can also cause underfitting, with the models assigning some of their probability mass to less probable configurations, as they need to be able to generalise over whole sequences. Using the mismatch string kernel – which counts the number of short segments shared between sequences – samples from the Variable-gram Topic models exhibit lower similarity to true pieces when compared to samples from the non-topic Dirichlet-VMM. On the other hand, the evaluation using the normalised mismatch kernel – which takes into account both the inter and the intra-sequence structure – indicates the Variable-gram Topic model as the best performer.

Overall, both evaluation objectives highlight the importance of dealing with novelty when modelling music. Each piece is a unique music idea, which means that new elements will emerge and be shared within a piece, which are not shared across pieces. In a prediction task this can be handled by allowing the models to adapt as we observe a new sequence. In a music generation task this is more difficult to accomplish as the only available information comes from what the model has already generated. In the topic models the differentiation between inter and intra-sequence structure comes from the fact that each piece has a unique distribution over topics. Therefore, a possible way of generating new pieces, would be to condition on the topic allocations for the training sequences in order to reduce the sampling noise coming from the unknown distribution over topics.

Chapter 6

Discussion

In this work we addressed the problem of learning a generative model for melody from a set of music pieces belonging to the same genre. This is a challenging problem for machine learning methods as melody exhibits complex temporal dependencies, which are further influenced by the realisation of several musical components, such as meter and rhythm. This kind of complexity is common in sequential data and is often dealt with by introducing domain knowledge, for instance through feature engineering or designing of models that reflect the specific structure of a problem. However, such approaches are often laborious and difficult to generalise across different datasets. Here we examined more generic methods which do not depend on specific musical knowledge and can easily be transferred to other music genres and other sequential data domains.

Therefore, our main contributions are in the design and evaluation of probabilistic models that can automatically learn the melodic structure from data sequences. We started by considering dictionary-based predictors in Chapter 3, which have previously been shown to produce state-of-the-art results in melody generation. The nature of these models is fairly simple, as they do not consider any latent variables. However, they can learn temporal dependencies of considerable complexity by parsing sequences into a lexicon of motifs, which are of variable length. In the melody modelling task this is a very powerful property as musical motifs can be seen as the building blocks of musical pieces.

Our first contribution is a novel form of the variable-length Markov model, the Dirichlet-VMM. This is an iteratively refined model that uses Bayesian agents

to leverage all available information, while remaining analytically tractable for inference and learning. The model handles the issue of smoothing by recursively tying the predictive probabilities of longer contexts to the statistics of shorter ones. This is a form of linear interpolation, where the weights for the lower and higher order statistics are automatically defined through a prior Dirichlet distribution. The tractability of the model is crucial, as it allows us to use the Dirichlet-VMM as the building block of the Variable-gram Topic model introduced in Chapter 5. Furthermore, in Section 3.3 we examined how to deal with novelty in unseen music pieces by considering models that combine global information from the training data with local information coming from the already processed part of a testing piece.

In Chapter 4 we considered a model based on the Restricted Boltzmann Machine for the melody modelling task. We examined whether music melody can be decomposed into a set of latent features being "on" or "off" in the same way that images can be decomposed into sets of edges and corners. Given the importance of motifs as the smallest structural elements of melody, we developed the Time Convolutional Restricted Boltzmann Machine, a fully undirected network, where each hidden unit receives input from a visible subsequence, rather than an individual time-step. Information is propagated in time by additionally connecting each visible time-step to multiple hidden time-steps. The weights are tied across time-steps, which allows us to use convolution to implement inference efficiently.

In Chapter 5 we introduced the Variable-gram Topic model, which couples the latent topic formalism with an expressive model of contextual information. The model employs the Dirichlet-VMM for the topic distribution over words, which captures the temporal structure by learning contexts of variable length that are indicative of the future. At the same time, the latent topics represent different music regimes, thus allowing us to model the different styles, tonalities and dynamics that occur in music. The Variable-gram Topic model allows us to distinguish between inter and intra-sequence structure, as every piece is associated with a unique distribution over topics, while the topics are shared across pieces of the same genre.

A final contribution of this thesis is in the context of evaluation. In Chapter 2 we presented a novel way of assessing model performance by employing the Maximum Mean Discrepancy and the mismatch string kernel to measure how

close sampled generations are to true data sequences. This evaluation helped us highlight different aspects of model behaviour compared to next-step prediction log-likelihood. For instance, in Chapter 3 we saw that the global-local combination models that use dynamic tree structures (adding new nodes as we observe the testing piece) had the highest log-likelihood during cross-validation, but turned out to be poor generative models, with synthesised samples often continually repeating a single motif. Furthermore, looking at the mismatch kernels allowed us to visualise the progress that a model has made and the types of data structure that it fails to adequately capture. For instance, the difference between inter and intra-piece structure in music pieces is higher than in samples from any of the models.

6.1 Conclusion

Throughout this thesis we used two evaluation objectives to compare model performance: the average next-step prediction log-likelihood of held-out data and the Maximum Mean Discrepancy between model samples and true data sequences computed using the mismatch string kernel to quantify the similarity between sequences. In this Section we summarise our findings, identify areas that remain open for future research and draw an overall conclusion regarding the appropriateness of the different models for the tasks of melody prediction and generation.

Table 6.1 shows the average next-step prediction log-likelihood of testing data under the different models examined in this thesis, where for each model type we have selected the best performing configuration. We first examine the models that do not consider any information coming from the testing data (top three sub-tables). All models outperform the empirical marginal distribution and the highest log-likelihood is achieved by the Variable-gram Topic model with the deeper tree structure (Variable-gram.De).

The Dirichlet-VMMs perform significantly better than the VMM trained using the Probabilistic Suffix Tree algorithm and additive smoothing, indicating that the iteratively refined inference procedure of the former makes better use of the available data. Furthermore, the Dirichlet-VMMs outperform the TC-RBMs, the

Table 6.1: Average next step prediction log-likelihood of the testing data under different models. For each model type, we use the 10 trained models from cross-validation and compute the average next-step prediction log-likelihood of the testing data (64 pieces) under each of these models. The mean and standard deviation are computed across the results from the 10 trained models.

Stationary Models	Mean $\pm$ St.Dev.
Empirical Marginal	-2.2217 ± 0.0009
VMM	-1.7437 ± 0.0029
Dirichlet-Bigram	-1.9171 ± 0.0019
Dirichlet-VMM.Sh	-1.6084 ± 0.0030
Dirichlet-VMM.De	-1.5663 ± 0.0040
TC-RBM, $N_h = 100$, 250 Samples	
TCRBM-100	-1.7635 ± 0.0040
TCRBM-100 Sparse	-1.7683 ± 0.0368
Topic Models – Update $S.1$, $K = 50$	
LDA	-2.0217 ± 0.0030
Topic Bigram	-1.7077 ± 0.0039
Topic Variable-gram.Sh	-1.5311 ± 0.0055
Topic Variable-gram.De	-1.5223 ± 0.0076
Dirichlet-VMM Global-Local Combinations	
GL_1-fixed ($.De$)	-1.4454 ± 0.0038
GL_1-dynamic ($.Sh$)	-1.0393 ± 0.0054
GL_2 ($.Sh$)	-1.4309 ± 0.0052
GL_3-fixed ($.De$)	-1.3072 ± 0.0021
GL_3-dynamic ($.De$)	-1.0905 ± 0.0012
Topic Models – Update $S.3$, $K = 50$	
LDA	-2.0214 ± 0.0030
Topic Bigram	-1.6971 ± 0.0036
Topic Variable-gram.Sh	-0.9518 ± 0.0032
Topic Variable-gram.De	-0.9658 ± 0.0040

topic Bigram and LDA, which stresses the importance of considering contextual information of variable order when modelling music. This remark is further supported by the fact that the VMM performs significantly better than the Dirichlet-Bigram and LDA, with the latter two considering first-order and no temporal dependencies, respectively. However, we note that the topic Bigram, which models first order dependencies within the latent topic formalism performs better than the VMM with additive smoothing.

With respect to Markovian models, our focus was mainly on dictionary based predictors, which have proven to be state-of-the-art methods for melody generation. The comparison of the VMM and the Dirichlet-VMM with more specific language models, based for instance on probabilistic grammar induction and Hidden Markov models is left for future work. We believe that these methods could be very powerful in the context of melody modelling. However, their performance is often dependent on good engineering of the production rules or the hidden state transitions, based on domain knowledge. Building such representations is somewhat divergent from the scope of this work, as we are mainly interested in methods that can automatically learn these from the data.

The TC-RBMs are inferior to the Dirichlet-VMM in next-step prediction. Their predictive performance is comparable to that of the VMM, although still not as good. Similar to the VMM, they perform better than the Dirichlet-Bigram and LDA, but are outperformed by the Bigram and the Variable-gram Topic models. The main difference of the TC-RBM to the Markovian models examined in this thesis, is that it learns a distributed representation of the input space. The VMM and Dirichlet-VMM effectively parse music into a lexicon of motifs observed in the training data, thus assigning structure to individual forms. On the other hand the TC-RBM maps out the structure into complementary components that can synthesise the forms observed in musical subsequences, typically resulting in a much larger set of possible configurations. The TC-RBM learns descriptive musical features, but needs to be modulated by, for instance, stacking TC-RBMs on top of each other. The higher layers can learn which of these features appear together on broader temporal scales. We believe that further development of deep TC-RBM architectures is an interesting direction for future research.

The topic models always perform better than their non-topic counterparts in the next-step prediction task, indicating that the inclusion of latent topics is beneficial

when modelling melody. In the prediction task using the deeper tree structure for the Variable-gram Topic model gives the highest log-likelihood, followed by the Variable-gram Topic model with the shallower tree, with the difference between the two being fairly small. This is an interesting issue when it comes to model selection in a real setting, as the deeper tree results in a computationally more expensive model, both in terms of time and storage. Therefore, the appropriate tree structure should be chosen having the requirements of the application to be developed in mind.

Examining the models that combine information from the training data with information coming from the already processed testing piece (bottom two sub-tables), we can see that the predictive performance improves substantially in the Variable-gram Topic models and the Dirichlet-VMM global-local combinations, but is only marginally better for the Bigram Topic model and LDA. This illustrates that the short order statistics of melody do not change significantly across different pieces, while, as expected, longer contexts can be used differently within each piece.

Moving on to our second evaluation objective, Table 6.2 shows the squared MMD between model samples and held-out data sequences using the mismatch and the normalised mismatch kernels for the models examined throughout the thesis. When using the mismatch kernel, the Dirichlet-VMMs perform significantly better than the rest of the models. The Variable-gram Topic models become comparable to the Dirichlet-VMMs when we constrain the sampling procedure by clamping the topic allocations to the testing data (bottom sub-table).

With the normalised mismatch kernel the Variable-gram Topic models are the winners, followed by the Bigram Topic model and the Dirichlet-VMMs. The ability of the topic models to get a different distribution over topics for each piece allows them to distinguish between inter and intra-sequence structure, which is a desirable property when modelling melody. This is reflected in the results of this evaluation, as the normalised mismatch kernel (which considers both the inter and intra-sequence similarities) favours the Variable-gram Topic models, while the results from the mismatch kernel support the Dirichlet-VMMs.

The results for the sparse TC-RBM vary a lot across folds, indicating that the sparse networks can get stuck to bad local optima, thus generating samples that

Table 6.2: Mean, standard deviation, minimum and maximum of the estimated squared MMD between testing sequences and samples generated from different models. Results for the $(4,1)$ and the normalised $(4,1)$ mismatch kernels.

	$(4,1)$ Mismatch Kernel			Normalised $(4,1)$ Mismatch Kernel		
	Mean $\pm$ St.Dev.	Min	Max	Mean $\pm$ St.Dev.	Min	Max
Uniform	411.69 ± 1.31	409.34	413.13	0.1657 ± 0.0011	0.1639	0.1668
TrainData	6.22 ± 4.58	1.33	16.71	0.0030 ± 0.0015	0.0002	0.0048
Stationary models						
EmpMarg	129.89 ± 10.13	116.04	147.08	0.0644 ± 0.0054	0.0543	0.0723
VMM	173.58 ± 8.56	159.81	187.84	0.0269 ± 0.0046	0.0200	0.0360
Dir-Bigram	91.81 ± 6.10	83.67	102.18	0.0373 ± 0.0024	0.0342	0.0407
Dir-VMM.Sh	9.20 ± 4.97	2.71	19.79	0.0257 ± 0.0046	0.0163	0.0342
Dir-VMM.De	10.04 ± 5.23	2.18	18.71	0.0256 ± 0.0034	0.0205	0.0296
GL_1-fixed	8.36 ± 5.40	1.54	21.10	0.0246 ± 0.0040	0.0163	0.0306
TC-RBM, $N_h = 100$						
TCRBM-100	48.45 ± 3.81	39.30	52.19	0.0369 ± 0.0042	0.0299	0.0423
TCRBM-100.sp	158.25 ± 92.25	40.08	307.62	0.0994 ± 0.0471	0.0455	0.1959
Topic Models, $K = 50$, Samples from prior						
Bigram.K_{50}	96.10 ± 18.09	66.33	123.07	0.0271 ± 0.0058	0.0182	0.0370
Vargram.$Sh.K_{50}$	30.63 ± 9.96	12.01	46.73	0.0163 ± 0.0029	0.0117	0.0216
Vargram.$De.K_{50}$	39.00 ± 15.16	25.91	71.41	0.0160 ± 0.0037	0.0106	0.0209
Topic Models, $K = 50$, Samples given topics						
LDA.K^*_{50}	82.56 ± 9.35	70.85	101.67	0.0399 ± 0.0046	0.0338	0.0464
Bigram.K^*_{50}	52.31 ± 4.45	44.29	59.73	0.0170 ± 0.0025	0.0131	0.0204
Vargram.$Sh.K^*_{50}$	9.45 ± 5.21	2.95	20.03	0.0119 ± 0.0032	0.0084	0.0177
Vargram.$De.K^*_{50}$	16.96 ± 9.36	3.40	32.57	0.0123 ± 0.0040	0.0047	0.0194

are very similar to each other, but have a low level of similarity to true sequences. The non-sparse TC-RBM perform comparably to the topic Variable-grams when using the mismatch kernel, but are considerably worse when using the normalised kernel, as, similar to the Dirichlet-VMMs, they are unable to distinguish between inter and intra-sequence structure.

The MMD based evaluation is dependent on the notion of similarity induced by the kernel function. The mismatch kernels helps us identify whether sequences share many short substructures, but gives no indication of the types of substructures that are shared, nor on whether these are globally coherent. For instance, across a genre we can expect to have some very frequent motifs, but within a music piece we also have music segments that are unique or very rare. The Dirichlet-VMMs tend to generate the more frequent patterns, resulting in samples that have high similarity with many of the true sequences, but also with each other. On the other hand, the Variable-gram Topic models are trained to generalise over whole sequences leading to samples that are distinguishable from each other, but also have a lower overall level of similarity to the true music pieces. Designing kernel functions that can represent different notions of similarity between melodic sequences is also deferred for future research.

Overall, we would argue that both evaluations indicate that the Variable-gram Topic model is best suited for the melody modelling task. The model addresses three key structural elements of melody: contextual information – modelled through the Dirichlet-VMM, non-stationarity within a piece – modelled with the inclusion of latent topics, and uniqueness of each piece – modelled by having a different distribution over topics for each sequence. Our cross-validation experiments in Section 5.3 suggest that model performance depends on our selection of the hyper-parameters α and β for the prior Dirichlet distributions and therefore these should be properly validated. Model performance is very similar when we consider the shallow and the deep tree structures and therefore the meta-parameters that control the structure of the tree could be decided based on the available computational resources.

An important point we need to consider when modelling melody is how we deal with novelty. If we are interested in prediction, we can account for novelty by allowing a model to consider information coming from the new testing piece. We saw both in the case of the Dirichlet-VMM – with the global-local combinations –

and in the case of the topic models – with update schemes that re-estimate the distributions over words during prediction – that using information coming from the testing piece significantly improves performance. However, if we are interested in generation, i.e. sampling new music sequences that respect a musical style, then such information is not available. Allowing the model to learn and adapt to previously sampled points can sometimes help, as is the case of the GL_1-fixed Dirichlet-VMM. In the topic models a possible generation method is to consider the topic allocations for the training pieces and sample new notes conditioned on these.

Our two evaluation objectives — prediction log-likelihood and Maximum Mean Discrepancy — often gave different answers to our comparative analyses of the models. An interesting direction for future research is to construct metrics that assess model performance through human evaluation and to further examine whether any of the two evaluation objectives used in this work correlates well with quality as evaluated by humans. Human evaluation could be in the form of a Turing test, where a human judge listens to the melody of a piece from a human composer and to a synthesised sample from a model and decides which of the two is the human composition. Alternatively, we could construct evaluation systems similar to the ones used in the field of music information retrieval (MIR). For instance, we could ask listeners to rate model samples according to their resemblance to a musical genre. Finally, we could ask human experts to assess the musicality of model samples by rating different qualities of melodic synthesis, for example, how tuneful is a sample, whether it has global coherence and whether it respects metrical boundaries.

In the remainder of this discussion, we visually inspect synthesised samples from the different models examined in this thesis and compare them to true data sequences. We display music sheets of melodic sequences in Figures 6.1–6.5. To engrave the melody we used the `LilyPond` software, `version 2.16.2`. In model samples, we use the G-major key signature, which has one sharp, $F^{\#}$. The D-major key signature has two sharps, $F^{\#}$ and $C^{\#}$, so we can expect to see some $C^{\#}$ sharp notes explicitly annotated in the music sheets. The use of accidentals is rare in reels and thus sharp, flat and natural symbols in model samples indicate musical mistakes — with the exception of $C^{\#}$. Note that we only present the first 128 time-steps of each sequence, which corresponds to the first 16 musical

measures — remember that the meter is 4/4 and therefore each measure is 8 time-steps long.

Looking at the human compositions in Figure 6.1, it is obvious that repetition of phrases is a central part of the structure in all the tunes. Within every tune we can see phrases that often span 2 measures (or more) being repeated, typically after 2 or 4 measures. Short motifs are also repeated across pieces. For instance, in tunes 3 and 4 we can see the motif $F^{\#}GF^{\#}E$, although in tune 4 the $F^{\#}$ is a quarter note rather than an eighth.

Looking at the model samples in Figures 6.2–6.5 we can see that, as expected, none of the models is able to synthesise pieces that exhibit repetition of long phrases, and thus the overall structure of a sampled sequence is different to the overall structure of a data sequence. However, we can also see that all model samples are more structured and consonant compared to the sample from the uniform distribution and that in many model samples the local structures resemble those that occur in reels.

The VMM sample resembles the reel structure for the first 8 measures, with the movement in the melody line going up and down the scale notes, although we can also observe two accidentals. After the eighth measure the sample becomes more erratic with "jumps" inside the scale an more accidentals occurring. The Dirichlet-Bigram considers only the current note to sample the next note and we see that mistakes are more often after a quarter (or longer) note — as the current step is "continuation" and on each own has no information regarding the pitch value. However, we can see that even when we consider first order dependencies, the resulting sample resembles reels more closely than the sample from the empirical marginal distribution, indicating the importance of the context in melody.

Looking at Figure 6.3 we can see that the Dirichlet-VMMs exhibit phrases similar to the ones seen in reels, i.e. upward or downward movements through the scale notes. However, the samples do not have overall coherence and there are discontinuities between phrases. In both models (Dirichlet-VMM.Sh and Dirichlet-VMM.De), the sampling occasionally loses track of the melodic movement — resulting in one or two measures that are repetitive or out of tune — but often recovers and starts generating a different common phrase that resembles reels. For example,

in the Dirichlet-VMM.*De*, the melody changes after at the end of measure 5. The sample from the GL_1-fixed Dirichlet-VMM sounds more melodic — although this is to some extent down to personal opinion. In this model, counts from the already generated piece are added to the training data, which pushes the model to re-sample earlier configurations. In this sample we see repetition of short motifs — annotated by different coloured boxes — which create a musically pleasing outcome. However, these repetitions are not always consistent with the metrical boundaries of the melody and thus do not produce the global consistency that we find in reels.

The TC-RBM samples in Figure 6.4 are more irregular compared to the Dirichlet-VMM samples. The melody occasionally exhibits short phrases that resemble the reel style, but the samples sound mostly out of tune. Similarly, the samples from the topic variable-gram models — in which we sample both the topic allocation and the symbol — sound out of tune. The prior distribution over topics does not have any constraints as to which topics are found together in a piece. Therefore, under this sampling scheme we can have topics primarily used for tunes written in G-major and topics primarily used for tunes written in D-major mingled together. This can be seen in the sample from the Topic Variable-gram.*De* where we have $C^{\#}$ sharp followed by C in the 150th measure. From a modelling perspective it would be interesting to examine prior distributions that induce a covariance structure between topics.

In Figure 6.5 we can see samples from the topic models, in which we use the topic allocations from a data sequence and only sample the note symbol (word) given the topic and the context. We present samples from three models using the topic allocations from data tune 5 — shown at the top for comparison. The first thing we can observe is that by clamping the latent topics allocations to a real tune, the overall structure of the synthesised samples begins to resemble the structure we see in the data sequences. These samples exhibit downward and upward movements through the scale and within each sample we can see short phrases being repeated. However, we can also see that clamping topic allocations during sampling may be too constricting for synthesising *new* tunes. Both in the topic bigram and the topic variable-gram.*Sh* we can see that some phrases are exactly the same as in the data piece — denoted with blue and orange boxes respectively. This duplication from the data piece is more extensive in the case

of the topic bigram, as the use of longer, variable-length contexts in the topic variable-gram introduces more variation in the conditional distributions of the model. In the topic topic variable-gram.De — which considers longer contexts than the variable-gram.Sh — we can see that there is no exact duplication of sub-sequences from the data piece.

Overall, we see that none of the models captures the large-scale structure of melody sufficiently, and the synthesised sequences do not exhibit the repetition structure we observe in data sequences. Additionally, when we listen to model samples we notice the absence of metrical structure, which combined with the absence of phrasing results in sequences that lack global coherence. However, we also see that both the Dirichlet-VMM and the TC-RBM capture interesting short-term structure.

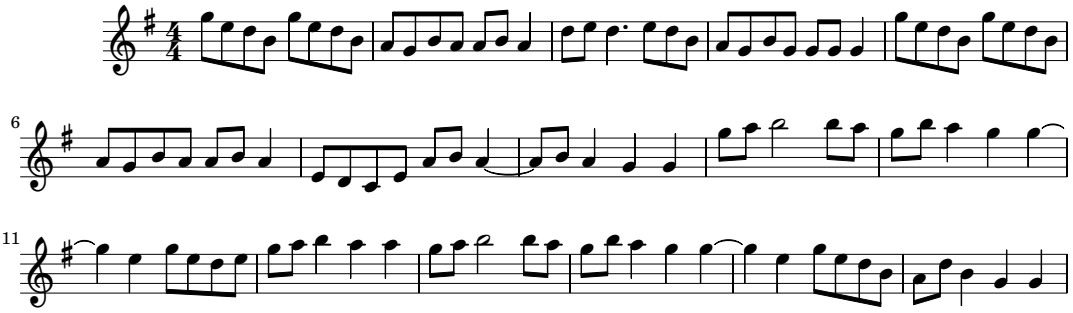
The Dirichlet-VMM is able to generate samples that contain the types of melodic phrases we see in reels. However, in order to acquire the large-scale phrasing structure that we see in reels, we need to also learn or hand-craft a latent structure that enforces repetition within each piece.

Sampling in the TC-RBM is more noisy and thus the generated sequences are more irregular compared to Dirichlet-VMM samples. The connectivity of the TC-RBM makes it more appropriate for in-filling tasks — where we sample time-steps between two subsequences — rather than forward sampling — where we only know the beginning of a sequence. Furthermore, stacking TC-RBMs could help modulate the appearance of the local features, as hidden time-steps at higher layers would effectively receive input from a much longer visible sub-sequence (through the hidden units in the lower layer).

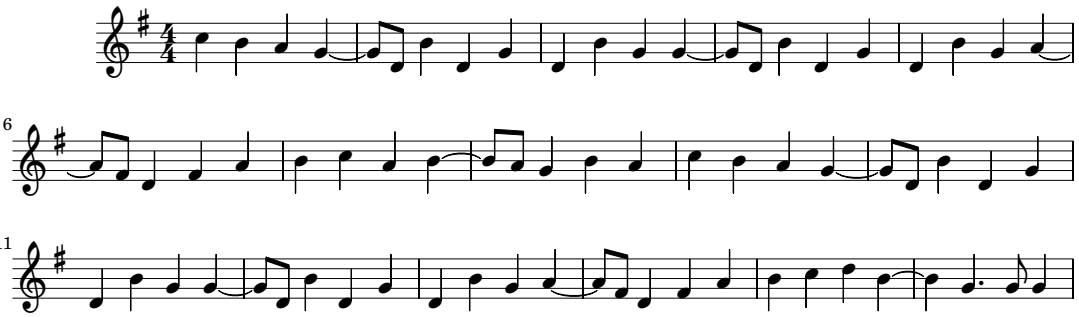
Finally, in the variable-gram topic models, if we constrain the sampling procedure using the latent topic allocations from data sequences, the resulting samples resemble the reel tunes more than samples from the rest of the models. However, we should be careful, as this restriction can lead to partial duplication of the data. The tradeoff between the sampling noise and the clamping restriction may be better dealt with if we consider topic models that can additionally learn correlation patterns of topic usage (see for example Blei and Lafferty (2007)). Additionally, it would be interesting to see how we can extend the topic models so that topic allocations respect the metrical boundaries of a piece, for instance, by having one

topic per music measure. Generally, if the objective is to synthesise new melodic pieces, incorporating metrical structure in the models is one of the first things we should consider in future research. A first step would be to additionally consider the velocity of each note event, besides its pitch and duration.

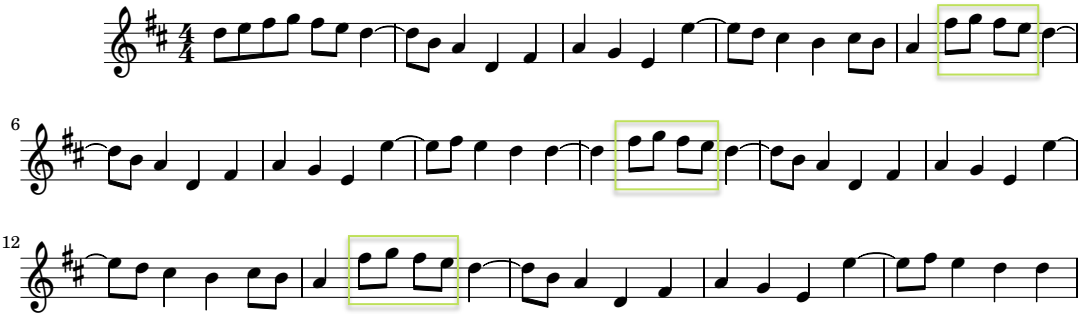
Data tune 1



Data tune 2



Data tune 3



Data tune 4

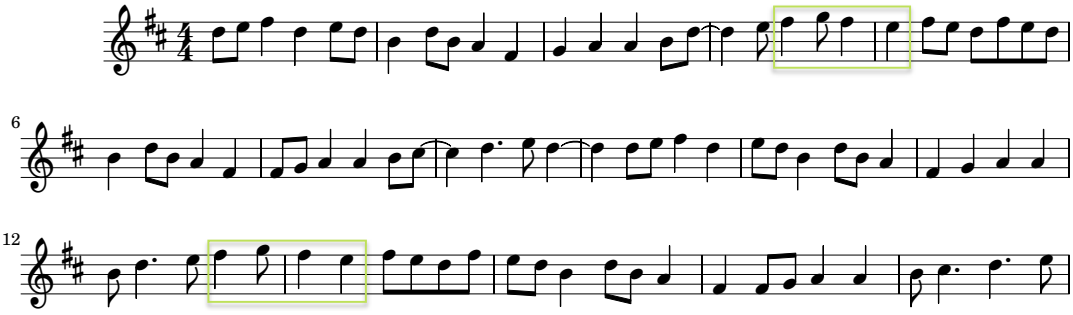


Figure 6.1: Melody of human composed tunes represented on the stave.

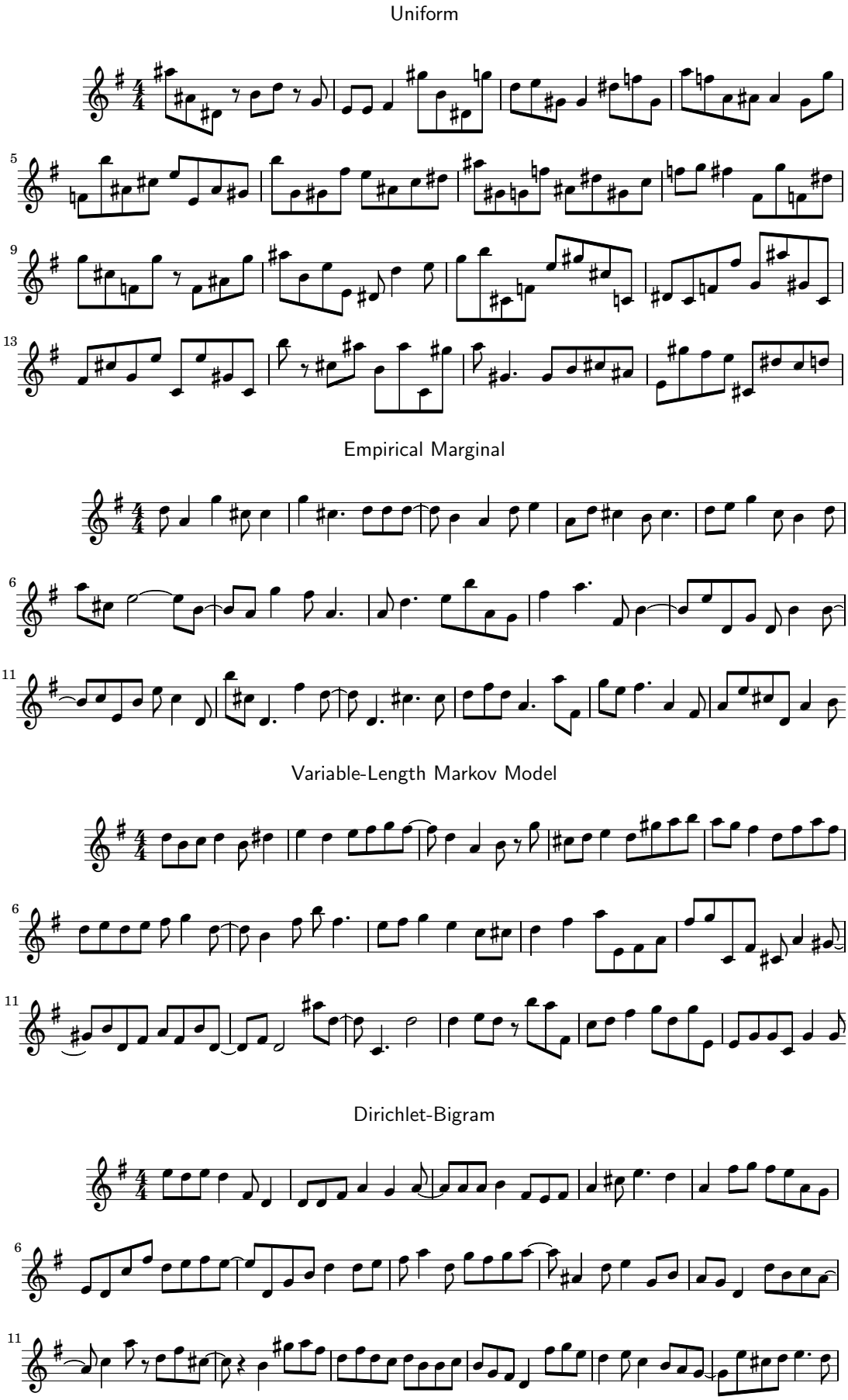


Figure 6.2: Melody of samples from different models represented on the stave.

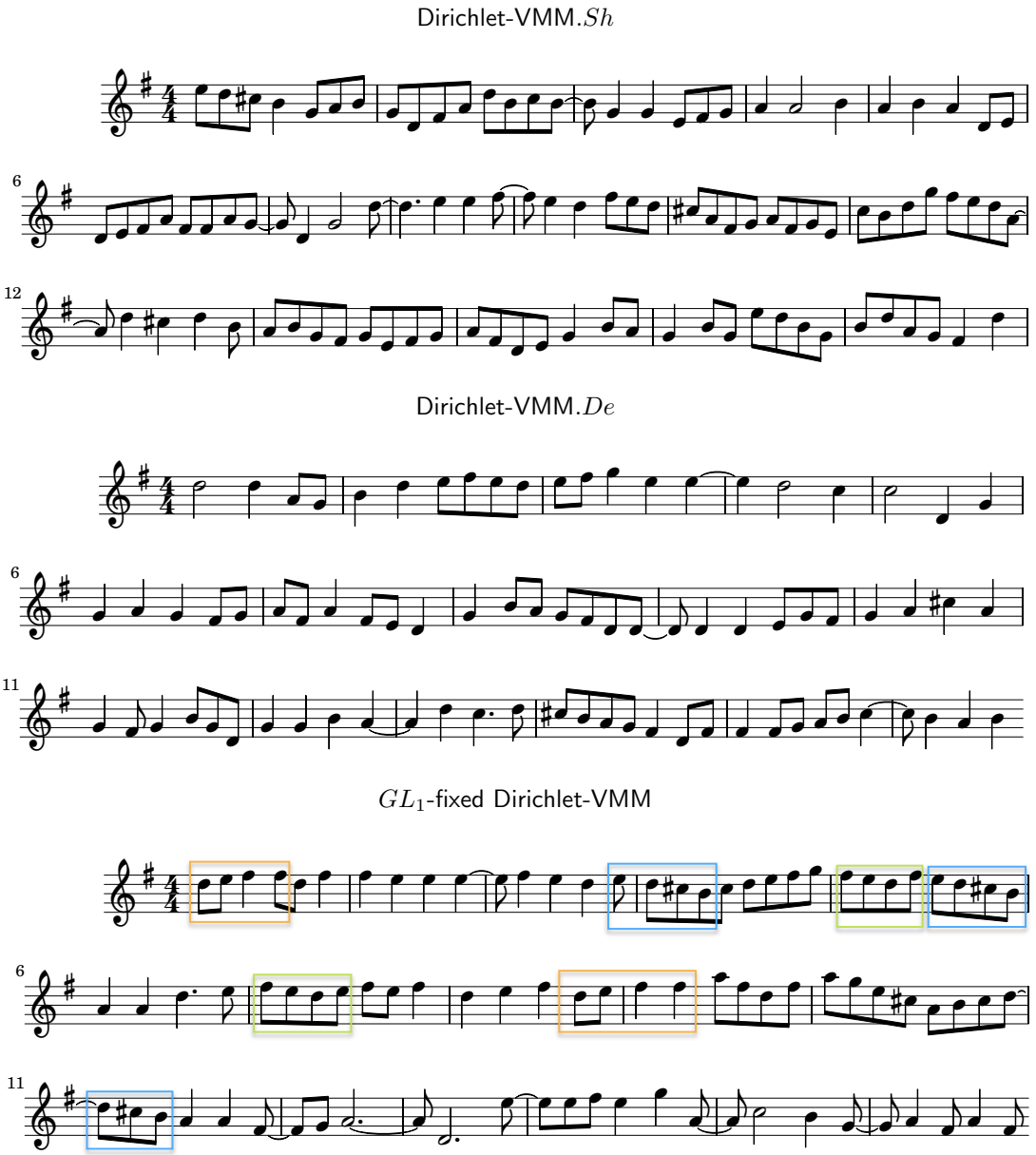
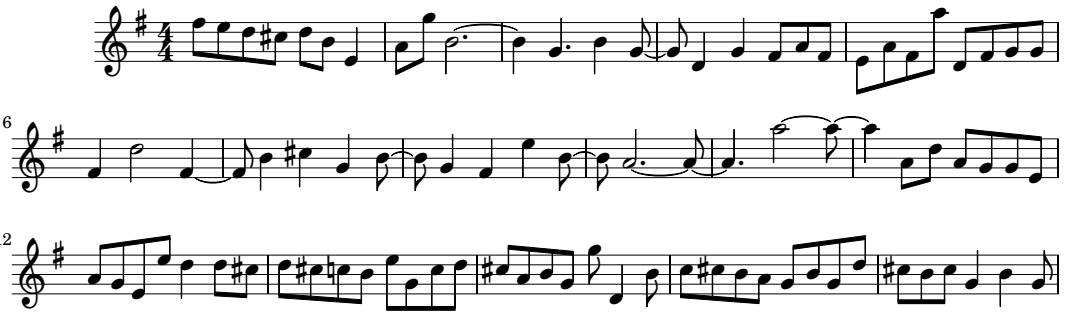
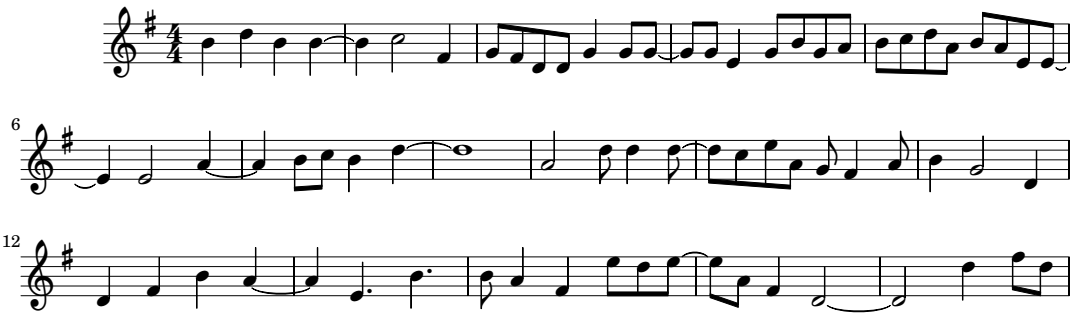


Figure 6.3: Melody of samples from different models represented on the stave.

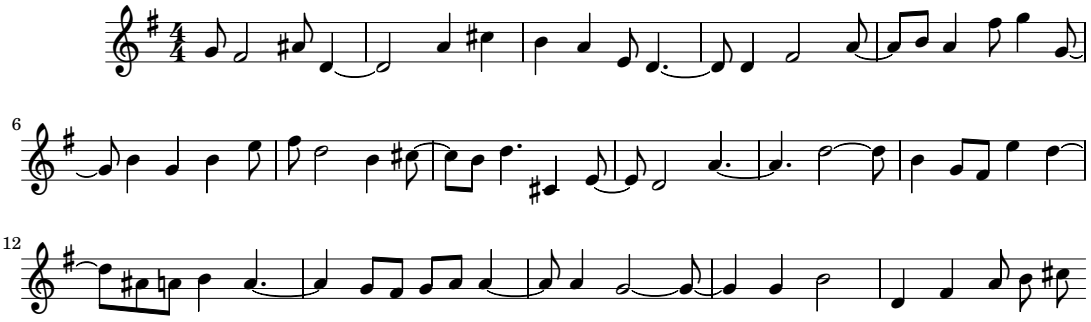
TC-RBM 100



TC-RBM sparse 100



Topic Variable-gram.$Sh.K_{50}$ – Sampling from the prior



Topic Variable-gram.$De.K_{50}$ – Sampling from the prior

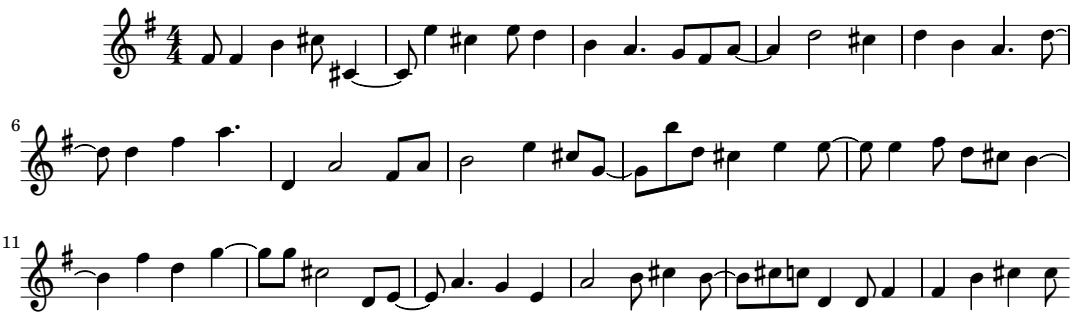


Figure 6.4: Melody of samples from different models represented on the stave.

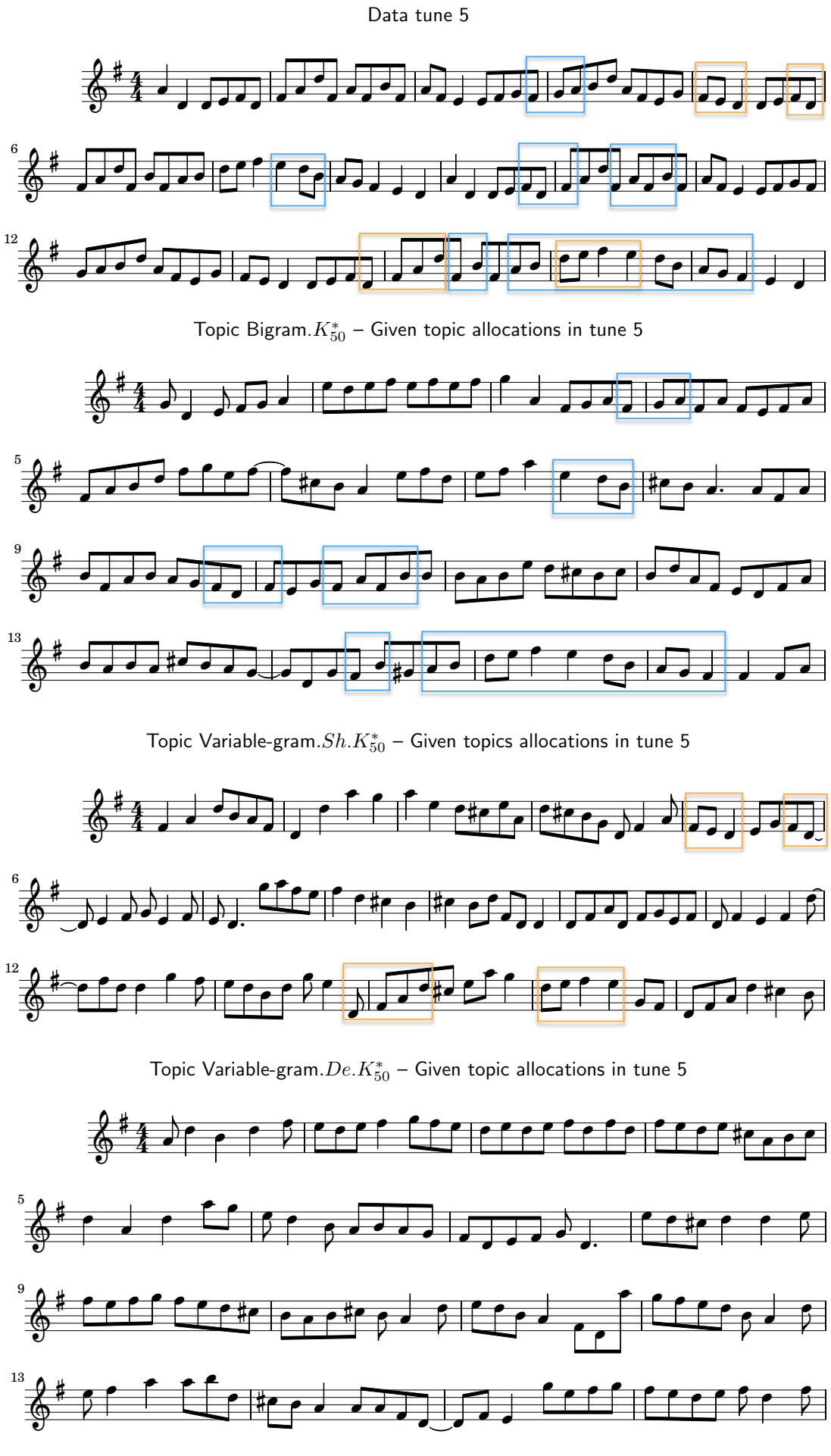


Figure 6.5: Melody of samples from different models represented on the stave.

Appendix A

Collapsed Gibbs Sampling in the Variable-gram Topic Model

In this Chapter we show how we can integrate out the parameters, $\mathbf{\Phi}$ and $\mathbf{\Theta}$, of the categorical distributions over words and topics, respectively, in the variable-gram topic model. Our goal is to infer the latent topic assignment for each word of every document and we do this through a collapsed Gibbs sampling procedure, where we eliminate the variables $\mathbf{\Phi}$ and $\mathbf{\Theta}$ and only sample the latent topics. This is a Rao-Blackwellisation (Casella and Robert, 1996) of the Gibbs sampler, where the variables $\mathbf{\Phi}$ and $\mathbf{\Theta}$ are marginalised out during sampling and are only tracked through their sufficient statistics. This is advantageous for two reasons. Firstly, the eliminated variables are effectively re-estimated with each sampling step, instead of once after a full pass over the data, which can be compelling at early stages. Secondly, the fact that the variables are tracked through their sufficient statistics, instead of being explicitly sampled, allows us to use the inference procedure of the Dirichlet-VMM to update the beliefs of the agents in each topic, as we will see below.

First, we derive a closed form expression for sampling a single topic assignment given the data, the remaining topic assignments and the hyperparameters, α, $\mathbf{n}$, β and $\{\mathbf{m}_{j,k}\}$. Then, we give a description of the complete inference procedure, which uses a Gibbs sampler to infer the latent topics and the iterative inference procedure of the Dirichlet-VMM to re-estimate the $\{\mathbf{m}_{j,k}\}$'s.

A.1 Model Definition

Let D be the number of documents, K the number of available topics, $\|\Sigma\|$ the number of available words in the vocabulary and J the number of leaf nodes in the Dirichlet-VMM tree, i.e. the number of possible contexts. Let $d \in \{1, \ldots, D\}$ index documents, $k \in \{1, \ldots, K\}$ index topics, $i \in \{1, \ldots, \|\Sigma\|\}$ index words from the vocabulary and $j \in \{1, \ldots, J\}$ index contexts. Additionally, let $t \in \{1, \ldots, T_d\}$ index the tokens from document d, where T_d is the number of words in the d-th document. Then we have

$$\boldsymbol{\theta}_d \sim \text{Dirichlet}(\alpha \mathbf{n}), \quad P(\boldsymbol{\Theta}|\alpha\mathbf{n}) = \prod_{d=1}^{D} \text{Dirichlet}(\boldsymbol{\theta}_d|\alpha\mathbf{n}) \ , \tag{A.1}$$

where we use a symmetric Dirichlet distribution with concentration parameter α and base measure $\mathbf{n} = [n_1 \ldots n_K]^\mathsf{T}$, with $n_1 = \ldots = n_K = \frac{1}{K}$ and $\sum_{k=1}^{K} n_k = 1$.

$$z_{t,d} \sim \text{Categorical}(\boldsymbol{\theta}_d) \ , \tag{A.2}$$

where $z_{t,d}$ is the latent topic assignment for t-th token in document d.

$$\boldsymbol{\phi}_{j,k} \sim \text{Dirichlet}(\beta \mathbf{m}_{j,k}), \quad P(\boldsymbol{\Phi}|\beta\{\mathbf{m}_{j,k}\}) = \prod_{k=1}^{K} \prod_{j=1}^{J} \text{Dirichlet}(\boldsymbol{\phi}_{j,k}|\beta\mathbf{m}_{j,k}) \ , \tag{A.3}$$

where β is the concentration parameter of the Dirichlet distribution and the $\mathbf{m}_{j,k}$'s are defined by the iterative inference procedure in the Dirichlet-VMM tree of the k-th topic, with $\mathbf{m}_{j,k} = \hat{\boldsymbol{\phi}}_{\mathbf{pa}(j),k}$. More specifically, $\{\mathbf{m}_{j,k}\}$ is the set of measures for the prior Dirichlet distributions of the agents sitting at the leaf nodes of each tree. Remember that the prior measures in the Dirichlet-VMM are estimated recursively starting at the root node and propagating relevant information in a top-down fashion. In the variable-gram topic model we are only interested in the conditional probability distributions identifying the leaf nodes, as these define the probability of the observed sequences under the model. In the following derivation we assume that an initial estimate of the $\{\mathbf{m}_{j,k}\}$'s is given and in Section A.4 we explain how these are re-estimated during Gibbs sampling.

Given the $\{\boldsymbol{\phi}_{j,k}\}$'s for all J leaf nodes in all K trees, we have

$$x_{t,d} \sim \text{Categorical}(\boldsymbol{\phi}_{\mathbf{c}_{t,d},z_{t,d}}) \ , \tag{A.4}$$

where $x_{t,d}$ and $\mathbf{c}_{t,d}$ are the word and the corresponding context (preceding observations) for token t in document d, respectively.

The joint probability of the data $\mathbf{X}$, the corresponding latent topic assignments, $\mathbf{Z}$, and the model parameters, $\mathbf{\Phi}$ and $\mathbf{\Theta}$, given hyperparameters α, $\mathbf{n}$, β and an estimate for the $\{\mathbf{m}_{j,k}\}$'s is given by

$$P(\mathbf{X}, \mathbf{Z}, \mathbf{\Phi}, \Theta|\alpha\mathbf{n}, \beta\{\mathbf{m}_{j,k}\}) = \prod_{d=1}^{D} P(\boldsymbol{\theta}_d|\alpha\mathbf{n}) \times \prod_{k=1}^{K}\prod_{j=1}^{J} P(\boldsymbol{\phi}_{j,k}|\beta\mathbf{m}_{j,k}) \times \prod_{d=1}^{D}\prod_{t=1}^{T_d} P(z_{t,d}|\boldsymbol{\theta}_d)P(x_{t,d}|\boldsymbol{\phi}_{\mathbf{c}_{t,d},z_{t,d}}). \tag{A.5}$$

A.2 Rao-Blackwellisation

For the Gibbs sampler we need to compute the probability of each topic assignment $z_{t,d}$, given the data $\mathbf{X}$, all other topic assignments, $\mathbf{Z}_{-(t,d)}$, the hyperparameters, α, $\mathbf{n}$, β and our current estimate of the $\{\mathbf{m}_{j,k}\}$'s. We have

$$P(z_{t,d}|\mathbf{Z}_{-(t,d)}, \mathbf{X}, \alpha\mathbf{n}, \beta\{\mathbf{m}_{j,k}\}) = \frac{P(\mathbf{Z}, \mathbf{X}|\alpha\mathbf{n}, \beta\{\mathbf{m}_{j,k}\})}{P(\mathbf{Z}_{-(t,d)}, \mathbf{X}|\alpha\mathbf{n}, \beta\{\mathbf{m}_{j,k}\})} \tag{A.6}$$

$$\propto P(\mathbf{Z}, \mathbf{X}|\alpha\mathbf{n}, \beta\{\mathbf{m}_{j,k}\}) \ , \tag{A.7}$$

where we have used the product rule of probability in (A.6) and we have dropped the denominator, which does not depend on $z_{t,d}$, in (A.7).

Therefore, the probability we wish to compute is proportional to the joint probability of the data, $\mathbf{X}$ and the latent topic assignments, $\mathbf{Z}$, given the model hyperparameters. Using the sum rule of probability and the model definition in (A.5) we have

$$\begin{aligned} P(\mathbf{Z}, \mathbf{X}|\alpha\mathbf{n}, \beta\{\mathbf{m}_{j,k}\}) &= \int\int P(\mathbf{X}, \mathbf{Z}, \mathbf{\Phi}, \Theta|\alpha\mathbf{n}, \beta\{\mathbf{m}_{j,k}\})\mathrm{d}\mathbf{\Phi}\mathrm{d}\mathbf{\Theta} \\ &= \int\int \prod_{d=1}^{D} P(\boldsymbol{\theta}_d|\alpha\mathbf{n}) \times \prod_{k=1}^{K}\prod_{j=1}^{J} P(\boldsymbol{\phi}_{j,k}|\beta\mathbf{m}_{j,k}) \\ &\times \prod_{d=1}^{D}\prod_{t=1}^{T_d} P(z_{t,d}|\boldsymbol{\theta}_d) \times \prod_{d=1}^{D}\prod_{t=1}^{T_d} P(x_{t,d}|\boldsymbol{\phi}_{\mathbf{c}_{t,d},z_{t,d}})\mathrm{d}\mathbf{\Phi}\mathrm{d}\mathbf{\Theta} \ . \end{aligned} \tag{A.8}$$

Next, we separate the integrals with respect to the variable being integrated

$$
\begin{aligned}
A.8 = &\int \prod_{d=1}^{D} \left(P(\boldsymbol{\theta}_d|\alpha\mathbf{n}) \prod_{t=1}^{T_d} P(z_{t,d}|\boldsymbol{\theta}_d) \right) \mathrm{d}\boldsymbol{\theta}_1 \dots \mathrm{d}\boldsymbol{\theta}_D \\
&\times \int \prod_{k=1}^{K} \prod_{j=1}^{J} P(\boldsymbol{\phi}_{j,k}|\beta\mathbf{m}_{j,k}) \prod_{d=1}^{D} \prod_{t=1}^{T_d} P(x_{t,d}|\boldsymbol{\phi}_{\mathbf{c}_{t,d},z_{t,d}}) \mathrm{d}\boldsymbol{\phi}_{1,1} \dots \mathrm{d}\boldsymbol{\phi}_{J,K} \qquad \text{(A.9)} \\
&= \prod_{d=1}^{D} \int P(\boldsymbol{\theta}_d|\alpha\mathbf{n}) \prod_{t=1}^{T_d} P(z_{t,d}|\boldsymbol{\theta}_d) \mathrm{d}\boldsymbol{\theta}_d \\
&\times \prod_{k=1}^{K} \prod_{j=1}^{J} \int P(\boldsymbol{\phi}_{j,k}|\beta\mathbf{m}_{j,k}) \prod_{d=1}^{D} \prod_{t=1}^{T_d} P(x_{t,d}|\boldsymbol{\phi}_{\mathbf{c}_{t,d},z_{t,d}}) \mathrm{d}\boldsymbol{\phi}_{j,k} \ . \qquad \text{(A.10)}
\end{aligned}
$$

Let

- N be the total number of tokens across all documents,
- $N_d = T_d$ be the number of tokens in document d, with $\sum_{d=1}^{D} N_d = N$,
- $N_{k|d}$ be the number of times topic k has been assigned to document d, with $\sum_{k=1}^{K} N_{k|d} = N_d$,
- $N_{j,k}$ be the number of times topic k has been assigned to any word preceded by context $\mathbf{c}_j$ across all documents, with $\sum_{j=1}^{J} \sum_{k=1}^{K} N_{j,k} = N$, and
- $N_{i|j,k}$ be the number of times topic k has been assigned to word i preceded by context $\mathbf{c}_j$ across all documents, with $\sum_{i=1}^{\|\Sigma\|} N_{i|j,k} = N_{j,k}$.

Now, we can replace the density functions in (A.10) using the definitions of the Dirichlet and the categorical distributions. We have

$$
\begin{aligned}
A.10 = &\prod_{d=1}^{D} \int \frac{\Gamma(\alpha)}{\prod_{k=1}^{K} \Gamma(\alpha n_k)} \prod_{k=1}^{K} \theta_{k|d}^{\alpha n_k - 1} \prod_{k=1}^{K} \theta_{k|d}^{N_{k|d}} \mathrm{d}\boldsymbol{\theta}_d \\
&\times \prod_{k=1}^{K} \prod_{j=1}^{J} \int \frac{\Gamma(\beta)}{\prod_{i=1}^{\|\Sigma\|} \Gamma(\beta m_{i|j,k})} \prod_{i=1}^{\|\Sigma\|} \phi_{i|j,k}^{\beta m_{i|j,k} - 1} \prod_{i=1}^{\|\Sigma\|} \phi_{i|j,k}^{N_{i|j,k}} \mathrm{d}\boldsymbol{\phi}_{j,k} \qquad \text{(A.11)} \\
&= \prod_{d=1}^{D} \int \frac{\Gamma(\alpha)}{\prod_{k=1}^{K} \Gamma(\alpha n_k)} \prod_{k=1}^{K} \theta_{k|d}^{\alpha n_k + N_{k|d} - 1} \mathrm{d}\boldsymbol{\theta}_d \\
&\times \prod_{k=1}^{K} \prod_{j=1}^{J} \int \frac{\Gamma(\beta)}{\prod_{i=1}^{\|\Sigma\|} \Gamma(\beta m_{i|j,k})} \prod_{i=1}^{\|\Sigma\|} \phi_{i|j,k}^{\beta m_{i|j,k} + N_{i|jk} - 1} \mathrm{d}\boldsymbol{\phi}_{j,k} \ . \qquad \text{(A.12)}
\end{aligned}
$$

Note that the Γ functions are constant with respect to the variables being integrated and can be taken outside the integral. Additionally, we can introduce the

appropriate normalising constants inside the integrals by simultaneously dividing outside the integral, as follows

$$
\begin{aligned}
A.12 = \prod_{d=1}^{D} \Bigg[& \frac{\Gamma(\alpha)}{\prod_{k=1}^{K}\Gamma(\alpha n_k)} \frac{\prod_{k=1}^{K}\Gamma(\alpha n_k + N_{k|d})}{\Gamma(\alpha + N_d)} \\
& \times \int \frac{\Gamma(\alpha + N_d)}{\prod_{k=1}^{K}\Gamma(\alpha n_k + N_{k|d})} \prod_{k=1}^{K} \theta_{k|d}^{\alpha n_k + N_{k|d} - 1} \mathrm{d}\boldsymbol{\theta}_d \Bigg] \\
& \times \prod_{k=1}^{K}\prod_{j=1}^{J} \Bigg[\frac{\Gamma(\beta)}{\prod_{i=1}^{\|\Sigma\|}\Gamma(\beta m_{i|j,k})} \frac{\prod_{i=1}^{\|\Sigma\|}\Gamma(\beta m_{i|j,k} + N_{i|j,k})}{\Gamma(\beta + N_{j,k})} \\
& \times \int \frac{\Gamma(\beta + N_{j,k})}{\prod_{i=1}^{\|\Sigma\|}\Gamma(\beta m_{i|j,k} + N_{i|j,k})} \prod_{i=1}^{\|\Sigma\|} \phi_{i|j,k}^{\beta m_{i|j,k} + N_{i|jk} - 1} \mathrm{d}\boldsymbol{\phi}_{j,k} \Bigg] \; .
\end{aligned} \tag{A.13}
$$

The integrals are now over the entire support of the Dirichlet distributions and thus evaluate to 1. So, overall we have

$$
\boxed{
\begin{aligned}
& P(\mathbf{Z}, \mathbf{X} | \alpha\mathbf{n}, \beta\{\mathbf{m}_{j,k}\}) = \\
& \quad \prod_{d=1}^{D} \frac{\Gamma(\alpha)}{\prod_{k=1}^{K}\Gamma(\alpha n_k)} \frac{\prod_{k=1}^{K}\Gamma(\alpha n_k + N_{k|d})}{\Gamma(\alpha + N_d)} \\
& \times \prod_{k=1}^{K}\prod_{j=1}^{J} \frac{\Gamma(\beta)}{\prod_{i=1}^{\|\Sigma\|}\Gamma(\beta m_{i|j,k})} \frac{\prod_{i=1}^{\|\Sigma\|}\Gamma(\beta m_{i|j,k} + N_{i|j,k})}{\Gamma(\beta + N_{j,k})} \; .
\end{aligned}
} \tag{A.14}
$$

A.3 Gibbs Sampling

For the purpose of the Gibbs sampling algorithm, we can further simplify the expression in (A.14) by dropping out terms that do not depend on the current latent topic assignment, $z_{t,d}$, since we are only interested in proportionality. Starting from (A.14), we drop the Γ functions that only depend on the hyperparameters, split the remaining products to separate terms that depend on the current topic assignment and then drop any remaining Γ functions that are constant with

respect to the current topic assignment.

$$
\begin{aligned}
&P(z_{t,d}|\mathbf{Z}_{-(t,d)}, \mathbf{X}, \alpha\mathbf{n}, \beta\{\mathbf{m}_{j,k}\}) \\
&\propto \prod_{d=1}^{D} \frac{\prod_{k=1}^{K} \Gamma(\alpha n_k + N_{k|d})}{\Gamma(\alpha + N_d)} \times \prod_{k=1}^{K}\prod_{j=1}^{J} \frac{\prod_{i=1}^{\|\Sigma\|} \Gamma(\beta m_{i|j,k} + N_{i|j,k})}{\Gamma(\beta + N_{j,k})} \qquad \text{(A.15)} \\
&= \prod_{d' \neq d} \frac{\prod_{k=1}^{K} \Gamma(\alpha n_k + N_{k|d'})}{\Gamma(\alpha + N_{d'})} \times \frac{\prod_{k=1}^{K} \Gamma(\alpha n_k + N_{k|d})}{\Gamma(\alpha + N_d)} \\
&\times \prod_{k=1}^{K} \prod_{j' \neq \mathbf{c}_{t,d}} \frac{\prod_{i=1}^{\|\Sigma\|} \Gamma(\beta m_{i|j',k} + N_{i|j',k})}{\Gamma(\beta + N_{j',k})} \\
&\times \prod_{k=1}^{K} \frac{\prod_{i' \neq x_{t,d}} \Gamma(\beta m_{i'|\mathbf{c}_{t,d},k} + N_{i'|\mathbf{c}_{t,d},k}) \times \Gamma(\beta m_{x_{t,d}|\mathbf{c}_{t,d},k} + N_{x_{t,d}|\mathbf{c}_{t,d},k})}{\Gamma(\beta + N_{\mathbf{c}_{t,d},k})} \qquad \text{(A.16)} \\
&\propto \frac{\prod_{k=1}^{K} \Gamma(\alpha n_k + N_{k|d})}{\Gamma(\alpha + N_d)} \times \prod_{k=1}^{K} \frac{\Gamma(\beta m_{x_{t,d}|\mathbf{c}_{t,d},k} + N_{x_{t,d}|\mathbf{c}_{t,d},k})}{\Gamma(\beta + N_{\mathbf{c}_{t,d},k})} \ . \qquad \text{(A.17)}
\end{aligned}
$$

Note that going from (A.16) to (A.17) we have assumed that the structure of the Dirichlet-VMM trees is the same for all the topics, i.e. the trees have the same leaf nodes. This means that if we observe context $\mathbf{c}_{t,d}$, then the counts for all other contexts remain unchanged irrespective of the topic assignment $z_{t,d}$.

Let $\left\{N_{[\cdot]}\right\}_{-(t,d)}$ with the appropriate subscript denote the different counts as above, having removed a single count for the word, the context and the topic assignment of the t-th token in document d. Then we can re-write (A.17) as follows with respect to the current topic assignment, $z_{t,d}$

$$
\begin{aligned}
A.17 = &\frac{\prod_{k' \neq z_{t,d}} \Gamma\left(\alpha n_{k'} + \left\{N_{k'|d}\right\}_{-(t,d)}\right) \times \Gamma\left(\alpha n_{z_{t,d}} + \left\{N_{z_{t,d}|d}\right\}_{-(t,d)} + 1\right)}{\Gamma\left(\alpha + \left\{N_d\right\}_{-(t,d)} + 1\right)} \\
&\times \prod_{k' \neq z_{t,d}} \frac{\Gamma\left(\beta m_{x_{t,d}|\mathbf{c}_{t,d},k'} + \left\{N_{x_{t,d}|\mathbf{c}_{t,d},k'}\right\}_{-(t,d)}\right)}{\Gamma\left(\beta + \left\{N_{\mathbf{c}_{t,d},k'}\right\}_{-(t,d)}\right)} \\
&\times \frac{\Gamma\left(\beta m_{x_{t,d}|\mathbf{c}_{t,d},z_{t,d}} + \left\{N_{x_{t,d}|\mathbf{c}_{t,d},z_{t,d}}\right\}_{-(t,d)} + 1\right)}{\Gamma\left(\beta + \left\{N_{\mathbf{c}_{t,d},z_{t,d}}\right\}_{-(t,d)} + 1\right)} \ . \qquad \text{(A.18)}
\end{aligned}
$$

Next, we use the recurrence relation of the Γ function, $\Gamma(y+1) = y\Gamma(y)$, which

allows us to merge the Γ functions into products across all topics k. We have

$$
\begin{aligned}
A.18 = & \frac{\prod_{k=1}^{K} \Gamma\left(\alpha n_k + \left\{N_{k|d}\right\}_{-(t,d)}\right) \times \left(\alpha n_{z_{t,d}} + \left\{N_{z_{t,d}|d}\right\}_{-(t,d)}\right)}{\Gamma\left(\alpha + \left\{N_d\right\}_{-(t,d)}\right) \times \left(\alpha + \left\{N_d\right\}_{-(t,d)}\right)} \\
& \times \prod_{k=1}^{K} \frac{\Gamma\left(\beta m_{x_{t,d}|\mathbf{c}_{t,d},k} + \left\{N_{x_{t,d}|\mathbf{c}_{t,d},k}\right\}_{-(t,d)}\right)}{\Gamma\left(\beta + \left\{N_{\mathbf{c}_{t,d},k}\right\}_{-(t,d)}\right)} \\
& \times \frac{\left(\beta m_{x_{t,d}|\mathbf{c}_{t,d},z_{t,d}} + \left\{N_{x_{t,d}|\mathbf{c}_{t,d},z_{t,d}}\right\}_{-(t,d)}\right)}{\left(\beta + \left\{N_{\mathbf{c}_{t,d},z_{t,d}}\right\}_{-(t,d)}\right)} \quad . \qquad \text{(A.19)}
\end{aligned}
$$

Again the Γ functions do not depend on the topic assignment $z_{t,d}$, so we can drop them to get

$$
\boxed{
\begin{aligned}
& P(z_{t,d}|\mathbf{Z}_{-(t,d)}, \mathbf{X}, \alpha\mathbf{n}, \beta\{\mathbf{m}_{j,k}\}) \propto \\
& \frac{\alpha n_{z_{t,d}} + \left\{N_{z_{t,d}|d}\right\}_{-(t,d)}}{\alpha + \left\{N_d\right\}_{-(t,d)}} \times \frac{\beta m_{x_{t,d}|\mathbf{c}_{t,d},z_{t,d}} + \left\{N_{x_{t,d}|\mathbf{c}_{t,d},z_{t,d}}\right\}_{-(t,d)}}{\beta + \left\{N_{\mathbf{c}_{t,d},z_{t,d}}\right\}_{-(t,d)}} \quad .
\end{aligned}
} \qquad \text{(A.20)}
$$

Note that the term $\alpha + \{N_d\}_{-(t,d)}$ can also be dropped, as it does not depend on the current topic assignment. However, from (A.20) it is easy to see that the Gibbs sampler keeps track of the sufficient statistics for the variables $\mathbf{\Phi}$ and $\mathbf{\Theta}$. Given a sampled topic assignment, $\mathbf{Z}$, we can estimate $\mathbf{\Phi}$ and $\mathbf{\Theta}$ as follows

$$
\begin{aligned}
\hat{\phi}_{i|j,k} &= P(x^* = i|\mathbf{c}^* = j, z^* = k, \mathbf{X}, \mathbf{Z}, \beta\mathbf{m}_{j,k}) \\
&= E_{P(\boldsymbol{\phi}_{j,k}|\mathbf{X},\mathbf{Z},\beta\mathbf{m}_{j,k})}\left[\phi_{i|j,k}\right] \\
&= \frac{\beta m_{i|j,k} + N_{i|j,k}}{\beta + N_{j,k}} \quad , \qquad \text{(A.21)} \\
\hat{\theta}_{k|d} &= P(z^* = k|\mathbf{X}, \mathbf{Z}, \alpha\mathbf{n}) \\
&= E_{P(\boldsymbol{\theta}_d|\mathbf{X},\mathbf{Z},\alpha\mathbf{n})}\left[\theta_{k|d}\right] \\
&= \frac{\alpha n_k + N_{k|d}}{\alpha + N_d} \quad . \qquad \text{(A.22)}
\end{aligned}
$$

As shown these values correspond to the predictive distributions over new words and new topics conditioned on the data, $\mathbf{X}$ and the latent topic assignments, $\mathbf{Z}$. An equivalent derivation for Latent Dirichlet Allocation can be found in (Carpenter, 2010).

A.4 Inference Procedure

To sample latent topics, we can normalise (A.20) by summing over all topics K and dividing as follows

$$P(z_{t,d}|\mathbf{Z}_{-(t,d)}, \mathbf{X}, \alpha\mathbf{n}, \beta\{\mathbf{m}_{j,k}\}) = \frac{\frac{\alpha n_{z_{t,d}} + \left\{N_{z_{t,d}|d}\right\}_{-(t,d)}}{\alpha + \{N_d\}_{-(t,d)}} \times \frac{\beta m_{x_{t,d}|\mathbf{c}_{t,d},z_{t,d}} + \left\{N_{x_{t,d}|\mathbf{c}_{t,d},z_{t,d}}\right\}_{-(t,d)}}{\beta + \left\{N_{\mathbf{c}_{t,d},z_{t,d}}\right\}_{-(t,d)}}}{\sum_{k=1}^{K}\left(\frac{\alpha n_k + \left\{N_{k|d}\right\}_{-(t,d)}}{\alpha + \{N_d\}_{-(t,d)}} \times \frac{\beta m_{x_{t,d}|\mathbf{c}_{t,d},k} + \left\{N_{x_{t,d}|\mathbf{c}_{t,d},k}\right\}_{-(t,d)}}{\beta + \left\{N_{\mathbf{c}_{t,d},k}\right\}_{-(t,d)}}\right)} \tag{A.23}$$

To perform inference in the variable-gram topic model, we randomly initialise all topic assignments and use the learned parameters of the Dirichlet-VMM to initialise the $\{\mathbf{m}_{j,k}\}$'s in all the topics. Then at each Gibbs cycle, we iterate over all tokens, subtract the current word and topic from the counts, use (A.23) to sample a new topic, and finally add the new topic and the corresponding word to the counts.

Every time we subtract or add counts, we also update the Dirichlet-VMM trees of the old and the new topic, respectively. In the Dirichlet-VMM the counts "sit" on every node with a conditional probability distribution satisfying the data, so for a given word-context pair, $(x, \mathbf{c})$, we need to update the counts at all nodes in the path from the leaf node with context $\mathbf{c}$ to the root. Note that these nodes have contexts of the form suffix($\mathbf{c}$). In order to reduce computation we do not perform the complete iterative inference procedure of the Dirichlet-VMM every time the counts change. Instead, we keep the updated counts at every node and estimate each $\mathbf{m}_{j,k}$ on the fly as required by the Gibbs sampler.

Note that for the prior distribution at the root node of each topic-tree we use a symmetric Dirichlet distribution with base measure $\boldsymbol{\phi}_{b,k} = \left[\phi_{1|b,k} \ldots \phi_{\|\Sigma\|\ |b,k}\right]^{\mathsf{T}}$, with $\phi_{1|b,k} = \ldots = \phi_{\|\Sigma\|\ |b,k} = \frac{1}{\|\Sigma\|}$ and $\sum_{i=1}^{\|\Sigma\|} \phi_{i|b,k} = 1$.

Bibliography

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, 9(1):147–169.

Allan, M. and Williams, C. K. I. (2005). Harmonising chorales by probabilistic inference. In Saul, L. K., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 17*, pages 25–32. MIT Press, Cambridge, MA.

Barrington, L., Oda, R., and Lanckriet, G. (2009). Smarter than genius? Human evaluation of music recommender systems. In *Proceedings of the 10th International Society for Music Information Retrieval Conference*, pages 357–362, Kobe, Japan.

Begleiter, R., El-Yaniv, R., and Yona, G. (2004). On prediction using variable order Markov models. *Journal of Artificial Intelligence Research*, 22:385–421.

Bengio, Y. (1996). Markovian models for sequential data. Technical Report 1049, Université de Montréal.

Bengio, Y. (2007). Learning deep architectures for AI. Technical Report 1312, Université de Montréal.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(1):281–305.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Blei, D. and Lafferty, J. (2007). A correlated topic model of Science. *The Annals of Applied Statistics*, 1(1):17–35.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Carpenter, B. (2010). Integrating out multinomial parameters in latent Dirichlet allocation and naive Bayes for collapsed Gibbs sampling. Technical report, LingPipe.

Casella, G. and Robert, C. P. (1996). Rao-Blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94.

Cemgil, A. T. (2004). *Bayesian Music Transcription*. PhD thesis, Radboud University of Nijmegen.

Chang, J., Boyd-Graber, J., Gerrish, S., Wang, C., and Blei, D. (2009). Reading tea leaves: How humans interpret topic models. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. K. I., and Culotta, A., editors, *Advances in Neural Information Processing Systems 22*, pages 288–296.

Chen, S. and Goodman, J. (1998). An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University.

Conklin, D. and Witten, I. H. (1995). Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73.

Downie, J. S. (2008). The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255.

Dubnov, S., Assayag, G., Lartillot, O., and Bejerano, G. (2003). Using machine-learning methods for musical style modeling. *IEEE Computer*, 36(10):73–80.

Eck, D. and Lapalme, J. (2008). Learning musical structure directly from sequences of music. Technical Report 1300, Université de Montreal.

Eck, D. and Schmidhuber, J. (2002). Learning the long-term structure of the Blues. In Dorronsoro, J. R., editor, *Proceedings of the International Conference on Artificial Neural Networks*, pages 284–289.

Eerola, T. and Toiviainen, P. (2004). *MIDI Toolbox: MATLAB Tools for Music Research*. University of Jyväskylä, Jyväskylä, Finland.

Franklin, J. A. (2006). Recurrent neural networks for music computation. *INFORMS Journal on Computing*, 18(3):321–338.

Ghahramani, Z. (2001). An introduction to hidden Markov models and Bayesian

networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(1):9–42.

Gretton, A., Borgwardt, K. M., Rasch, M., Schölkopf, B., and Smola, A. J. (2007). A kernel method for the two-sample-problem. In Schölkopf, B., Platt, J., and Hoffman, T., editors, *Advances in Neural Information Processing Systems 19*, pages 513–520. MIT Press, Cambridge, MA.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773.

Griffiths, T. L. and Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235.

Gruzd, A. A., Downie, J. S., Jones, M. C., and Lee, J. H. (2007). Evalutron 6000: collecting music relevance judgments. In *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, JCDL '07, pages 507–507, New York, NY, USA. ACM.

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

Hinton, G. E. (2010). A practical guide to training restricted Boltzmann machines. Technical Report TR 2010-003, Department of Computer Science, University of Toronto.

Hinton, G. E., Osindero, S., and Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

Lavrenko, V. and Pickens, J. (2003). Polyphonic music modeling with random fields. In Rowe, L. A., Vin, H. M., Plagemann, T., Shenoy, P. J., and Smith, J. R., editors, *Proceedings of the 11th ACM International Conference on Multimedia*, pages 120–129.

Lee, H., Ekanadham, C., and Ng, A. Y. (2008). Sparse deep belief net model for visual area V2. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems 20*, pages 873–880. MIT Press, Cambridge, MA.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2009). Convolutional deep

belief networks for scalable unsupervised learning of hierarchical representations. In Bottou, L. and Littman, M., editors, *Proceedings of the 26th International Conference on Machine Learning*. Omnipress.

Lerdahl, F. and Jackendoff, R. (1983). *A Generative Theory of Tonal Music*. The MIT Press, Cambridge, Massachusetts, London, England.

Leslie, C. S., Eskin, E., Cohen, A., Weston, J., and Noble, W. S. (2004). Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–476.

Mackay, D. J. C. and Peto, L. C. B. (1995). A hierarchical Dirichlet language model. *Natural Language Engineering*, 1(3):1–19.

Mozer, M. C. (1994). Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multiscale processing. *Connection Science*, 6(2-3):247–280.

Murray, I. and Salakhutdinov, R. (2009). Evaluating probabilities under high-dimensional latent variable models. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 1137–1144.

Norouzi, M., Ranjbar, M., and Mori, G. (2009). Stacks of convolutional restricted Boltzmann machines for shift-invariant feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2735–2742.

Paiement, J.-F. (2008). *Probabilistic Models for Music*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne (EPFL).

Paiement, J.-F., Bengio, S., and Eck, D. (2009a). Probabilistic models for melodic prediction. *Artificial Intelligence*, 173(14):1266–1274.

Paiement, J.-F., Eck, D., and Bengio, S. (2006). Probabilistic melodic harmonization. In *Proceedings of the 19th International Conference on Advances in Artificial Intelligence: Canadian Society for Computational Studies of Intelligence*, pages 218–229. Springer-Verlag.

Paiement, J.-F., Grandvalet, Y., and Bengio, S. (2009b). Predictive models for music. *Connection Science*, 21(2-3):253–272.

Pearce, M. and Wiggins, G. (2004). Improved methods for statistical modelling of monophonic music. *Journal of New Music Research*, 33(4):367–385.

Rabiner, L. R. and Juang, B. H. (1986). An introduction to hidden Markov models. *IEEE Acoustics, Speech and Signal Processing Magazine*, 3(1):4–16.

Ron, D., Singer, Y., and Tishby, N. (1996). The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25:117–149.

Salakhutdinov, R. and Hinton, G. (2009). Replicated softmax: an undirected topic model. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. K. I., and Culotta, A., editors, *Advances in Neural Information Processing Systems 22*, pages 1607–1614.

Salakhutdinov, R., Mnih, A., and Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering. In Ghahramani, Z., editor, *Proceedings of the 24th International Conference on Machine Learning*, pages 791–798. Omnipress.

Spiliopoulou, A. and Storkey, A. (2012). A topic model for melodic sequences. In Langford, J. and Pineau, J., editors, *Proceedings of the 29th International Conference on Machine Learning*, pages 1143–1150. Omnipress.

Spiliopoulou, A. and Storkey, A. J. (2011). Comparing probabilistic models for melodic sequences. In Gunopulos, D., Hofmann, T., Malerba, D., and Vazirgiannis, M., editors, *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 289–304. Springer.

Sriperumbudur, B. K., Fukumizu, K., Gretton, A., Lanckriet, G. R. G., and Schölkopf, B. (2009). On integral probability metrics, ϕ-divergences and binary classification. *arXiv*, 0901.2698.

Sriperumbudur, B. K., Gretton, A., Fukumizu, K., Schölkopf, B., and Lanckriet, G. R. (2010). Hilbert space embeddings and metrics on probability measures. *Journal of Machine Learning Research*, 11:1517–1561.

Sutskever, I. and Hinton, G. E. (2007). Learning multilevel distributed representations for high-dimensional sequences. In Meila, M. and Shen, X., editors,

Proceedings of the 11th International Conference on Artificial Intelligence and Statistics, pages 548–555.

Taylor, G. W. and Hinton, G. E. (2009). Factored conditional restricted Boltzmann machines for modeling motion style. In Bottou, L. and Littman, M., editors, *Proceedings of the 26th International Conference on Machine Learning*, pages 1025–1032. Omnipress.

Taylor, G. W., Hinton, G. E., and Roweis, S. T. (2007). Modeling human motion using binary latent variables. In Schölkopf, B., Platt, J., and Hoffman, T., editors, *Advances in Neural Information Processing Systems 19*, pages 1345–1352. MIT Press.

Teh, Y. W. (2006). A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 985–992.

Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. (2006). Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Teo, C. H. and Vishwanathan, S. V. N. (2006). Fast and space efficient string kernels using suffix arrays. In Cohen, W. and Moore, A., editors, *Proceedings of the 23nd International Conference on Machine Learning*, pages 929–936. Omnipress.

Turian, J., Shen, L., and Melamed, I. D. (2003). Evaluation of machine translation and its evaluation. In *Proceedings of the 9th Machine Translation Summit of the International Association for Machine Translation*, pages 386–393.

Wallach, H., Mimno, D., and McCallum, A. (2009a). Rethinking lda: Why priors matter. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. K. I., and Culotta, A., editors, *Advances in Neural Information Processing Systems 22*, pages 1973–1981.

Wallach, H. M. (2006). Topic modeling: Beyond bag-of-words. In Cohen, W. and Moore, A., editors, *Proceedings of the 23rd International Conference on Machine Learning*, pages 977–984. Omnipress.

Wallach, H. M. (2008). *Structured Topic Models for Language*. PhD thesis, University of Cambridge.

Wallach, H. M., Murray, I., Salakhutdinov, R., and Mimno, D. (2009b). Evaluation methods for topic models. In Bottou, L. and Littman, M., editors, *Proceedings of the 26th International Conference on Machine Learning*, pages 1105–1112. Omnipress.

Weiland, M., Smaill, A., and Nelson, P. (2005). Learning musical pitch structures with hierarchical hidden Markov models. Technical report, University of Edinburgh.

Welling, M., Rosen-Zvi, M., and Hinton, G. E. (2005). Exponential family harmoniums with an application to information retrieval. In Saul, L. K., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 17*, pages 1481–1488. MIT Press.

Wood, F., Archambeau, C., Gasthaus, J., James, L., and Teh, Y. W. (2009). A stochastic memoizer for sequence data. In Bottou, L. and Littman, M., editors, *Proceedings of the 26th International Conference on Machine Learning*, pages 1129–1136, Montreal. Omnipress.