

---

# On discovery and exploitation of temporal structure in data sets

---

A dissertation submitted in fulfilment of the degree of Doctor of Philosophy.

Tim Scarfe



Computer Learning Research Centre and Department of Computer Science

Royal Holloway, University of London

United Kingdom

2015

## **Declaration**

I, Tim Scarfe, hereby declare that this thesis and the work presented in it is entirely my own. Where I have consulted the work of others, this is always clearly stated.

Signed ..... (Tim Scarfe)

Date:

## **Supervisors**

Dr Yuri Kalnishkan and Professor Alexander Gammerman

## Abstract

This thesis explores temporal structure based on self-similarity in different contexts.

An efficient dynamic programming algorithm is presented which discovers temporal structures in music shows, obtains high quality results, and compares them to similar algorithms used in the literature. The program segments a self-similarity matrix given a cost function and a fixed number of homogeneous temporal structures to find. This is the initial approach we use to discover temporal structures in music data.

The use of a self-similarity matrix to visualize temporal structures is discussed in detail. Then the following question is explored; if similar temporal structures in other corpora existed; could forecasting algorithms be adapted to take advantage of them even if they were not known a priori?

Prediction with expert advice techniques are then introduced to exploit a priori unknown temporal structures of a similar configuration in an on-line configuration. Uni-variate Russian Stock Exchange options futures volatility corpora are used, which are highly interesting for on-line forecasting.

We experiment with merging together expert models which have been trained in some way to recognise temporal structures in corpora. The first types are kernel ridge regression models trained to be experts on particular regions in time, or untrained and given random sets of parameters which may work well on certain time regions. The other types of model used are parsimonious predictors which transform uni-variate financial data into elementary time series based on homogeneous vicinities of information in the side domain. Expert merging techniques are then used across these time series which produce a validation-free forecaster comparable to sliding kernel ridge regression.

## Acknowledgement

Thank you primarily to Ruth for putting up with me during the last 7 years of my studies.

My main supervisor Yuri Kalnishkan for countless hours of fascinating discussion and infinite patience. I also had the huge privilege of working with an exceptionally talented scientist; Wouter M. Koolen. My co-supervisor Alex Gammerman for encouragement and support.

James Smith, Valentina Fedorova, Alexey Chernov, Dmitry Adamskiy, Andrej Gregoric, Jasper Lyons and Marko Babic for discussions about my work and indeed a whole spectrum of leisure activities.

Mikael Lindgren and Denis Goncharov from [cuenation](http://www.cuenation.com)<sup>1</sup> for providing the music corpus and descriptions of temporal structures in music.

---

<sup>1</sup><http://www.cuenation.com>

# Contents

<b>1</b>	<b>Introduction</b>	<b>8</b>
1.1	Summary of Thesis . . . . .	8
1.2	Contributions and Organisation . . . . .	11
1.2.1	Music Segmentation Algorithm . . . . .	11
1.2.2	Merging Ridge Regression Models . . . . .	11
1.2.3	Merging Time Series Algorithm . . . . .	12
1.3	Materials . . . . .	12
1.4	Papers . . . . .	13
<b>2</b>	<b>Temporal Structures in Univariate Data</b>	<b>14</b>
2.1	Homogeneous Structures . . . . .	14
2.2	Literature Review . . . . .	16
<b>3</b>	<b>Discovering Temporal Structures In Music</b>	<b>22</b>
3.1	Electronic Dance Music . . . . .	23
3.2	Music Corpora . . . . .	25
3.3	Human Accuracy . . . . .	29
3.4	Data Handling . . . . .	30
3.4.1	Preprocessing . . . . .	30
3.4.2	Feature Extraction . . . . .	32
3.4.2.1	Music . . . . .	32

3.4.2.2	Self-Similarity Matrix . . . . .	36
3.4.2.3	Cost Matrices . . . . .	39
3.5	Computing Best Segmentation . . . . .	44
3.6	Confidence Intervals . . . . .	47
3.6.1	Posterior Marginal of Song Boundary . . . . .	47
3.6.2	Posterior Marginal of Song Position . . . . .	49
3.6.3	Confidence Measures . . . . .	49
3.6.3.1	Index (Order) . . . . .	49
3.6.3.2	Time . . . . .	50
3.7	Experiments . . . . .	50
3.7.1	Training Set . . . . .	50
3.7.2	Number Of Tracks Known A Priori . . . . .	51
3.7.2.1	Evaluation . . . . .	51
3.7.2.2	Finding The Best Parameters . . . . .	52
3.7.2.3	Results . . . . .	52
3.7.2.4	Confidence Interval Analysis . . . . .	54
3.7.3	Number Of Tracks Not Known A Priori . . . . .	54
3.7.3.1	Comparison of Methods For Segmentation . . . . .	56
3.7.3.2	Results . . . . .	56
<b>4</b>	<b>Forecasting Framework</b>	<b>68</b>
4.1	Online Protocol/Regression . . . . .	68
4.2	Kernel Ridge Regression . . . . .	69
4.2.0.3	Cholesky Decomposition . . . . .	71
4.2.0.4	An Identity . . . . .	72
4.2.1	Normalisation . . . . .	72
4.3	Growing Window . . . . .	73
4.4	Sliding Window . . . . .	74

4.5	Merging Decision Strategies . . . . .	75
4.5.1	Prediction Framework . . . . .	75
4.5.2	Prediction with Expert Advice . . . . .	76
4.5.3	Aggregating Algorithm . . . . .	77
4.5.4	Sleeping and Specialist Experts . . . . .	79
4.5.5	Switching Experts . . . . .	80
4.5.6	Tracking for Sleeping Experts . . . . .	83
4.5.7	Sleeping Specialist Experts Performance Bounds . . . . .	85
4.5.8	Markov Chains of Experts . . . . .	85
<b>5</b>	<b>RTSSE Corpora</b>	<b>88</b>
5.1	Implied Volatility Prediction . . . . .	88
5.1.1	Implied Volatility . . . . .	88
5.1.2	RTSSE Datasets . . . . .	90
5.2	Qualitative Analysis . . . . .	94
5.2.1	Time Progression . . . . .	94
5.2.2	Implied volatility Distribution . . . . .	94
5.2.3	Strike Distribution . . . . .	95
5.2.4	Self-Similarity Map With Predictive Regions . . . . .	96
5.3	Temporal Structures Discussed . . . . .	96
5.4	Exploiting Temporal Structures . . . . .	99
<b>6</b>	<b>On-line Forecasting With Specialist Experts</b>	<b>111</b>
6.0.1	On-line Kernel Ridge Regression . . . . .	111
6.1	Merging Temporal Dependencies . . . . .	115
6.1.1	Side-Domain Time Series Merging Algorithm . . . . .	122
6.1.2	Selecting the Range of Sizes . . . . .	124
6.1.3	Selecting Types of Vicinities . . . . .	125
6.2	Merging Spacial Dependencies . . . . .	126

---

6.2.1	Merging Fixed Region Experts . . . . .	126
6.2.2	Merging Lagged Region Experts . . . . .	128
6.2.3	Merging Ridge Models with Variable Window Sizes . . . . .	129
6.2.4	Stochastic Ridge Models . . . . .	129
6.2.4.1	Average Random Ridge Regression Algorithm . . . . .	130
6.2.4.2	Average Merged Random Ridge Regression . . . . .	130
6.2.4.3	Meta-merging . . . . .	130
6.3	Final Parameters & Results . . . . .	131
<b>7</b>	<b>Conclusions</b>	<b>141</b>
7.1	Discovering Temporal Structures In Music . . . . .	141
7.2	Forecasting on RTSSE Corpora . . . . .	143
7.2.1	Merging Temporal Models . . . . .	143
7.2.2	Merging Spacial Models . . . . .	145
7.3	Further Work . . . . .	148
	<b>Bibliography</b>	<b>150</b>
	<b>Appendices</b>	<b>165</b>
<b>A</b>	<b>Genetic Search Figures</b>	<b>166</b>

# Chapter 1

## Introduction

My inspiration to work on machine learning was first triggered by working on a 3d data visualisation project called VizZy back in 2004 for a company called Z/Yen<sup>1</sup>. The underlying algorithm was Vapnik's support vector machine classifier [36, 123, 124]. This triggered a fascination in machine learning and statistics that eventually culminated in this thesis.

I was greatly inspired by the work of Foote et al in his approach to the analysis of music and video scene data [44, 45, 46, 47, 49]. Foote segmented music data into small adjacent non-overlapping windows and built a two-dimensional similarity matrix using a trivial distance function (for example, the cosine distance). In an image-plot of this similarity matrix, temporal structures are clearly visible (see Figure 2.1 for an illustration of it applied to a music piece).

### 1.1 Summary of Thesis

In Chapter 3, Foote's work is replicated on a large corpus of music and an algorithm is described to find a fixed number of segments (temporal structures corresponding to songs in the music). The accuracy of these reconstructed temporal structures is

---

<sup>1</sup><http://www.zyen.com/>

compared to ground truth annotations captured by human domain experts.

A particular interest is discovering whether these temporal structures in music may also exist in financial options corpora and whether they could be exploited to provide practical improvements to forecasting algorithms such as time-series and on-line regression. This is the focus of Chapters 4,5 and 6. These practical improvements might for example, be improved; time-complexity, space-complexity, predictive performance or execution time. But more fundamentally; it would be desirable to have a forecasting algorithm that operates on a higher level of abstraction, such as one that considers temporal structures and adapts dynamically to them.

It was our intuition that other corpora would exhibit similar properties to music (for example; regions of self-similarity, contiguity, repetition).

One of the key concepts we explore is the idea of exactly how information from the past could help with forecasting in the future. If the nature of the data is changing over time; where different models predict well on different segments of the data – could an adaptive algorithm be designed which switches between these models?

We explore the question of whether the efficiency of forecasting algorithms could be improved. Let us presuppose that, in the context of an on-line regression algorithm, a contiguous region (or regime) of data is entered. This regime is not in flux and is self-similar. This would mean in theory that it is unnecessary to build a new model for the duration of this regime.

Firstly we confirm that these characteristic temporal structures do indeed exist in financial options data in Section 5.3 (and perhaps in Nature itself, although we do not explore this question directly). Instead of using the distance metrics advocated by Foote (cosine, euclidean), a regression algorithm is trained (which can also use information from the side domain) and retrieve its sum square loss on other regions in the corpus to create the self-similarity image for a domain outside of music (see Figures 5.6, 5.7 and 5.8).

We use the methodology of having forecasting models that correspond to evenly-

placed regions in time. And the construction of a self-similarity matrix where each cell corresponds to the predictive performance of one forecasting the other using a regression algorithm. Then, we optimally trace through the self-similarity matrix from beginning to end allowing different fixed numbers of switches to investigate the potential of a new switching region regression algorithm as an alternative to the usual sliding window regression approach.

We introduce the concept of specialist experts as a special case of the theory of expert evaluators (Section 4.5) after [26, 27] which essentially modifies Vovk’s Aggregating Algorithm [125] to allow experts to *go to sleep* or, in other words, abstain from making a prediction when it is unnecessary for them to do so. This is essential because expert regions cannot make predictions in any on-line scheme until their data region is in the past. We also use fixed share and variable share specialist expert algorithms (Herbster et al [64]) for comparison. These schemes are used to switch/merge dynamically between models that have been trained at different regions in time as described above.

As the region structure is not known a priori, we are aiming to beat the standard sliding window regression technique with some fixed number of switches allowed. This fixed number corresponds to the switching rate parameter present in the expert evaluator algorithms. The loss bound of the expert evaluator algorithms is strongly influenced by the switching rate parameter. So an expert evaluator algorithm may be able to beat the traditional sliding window on-line regression algorithm if it were also beatable with a small enough number of switches. For this to happen of course, old information would have to predict the future well. The traditional on-line sliding window approach to regression builds a new model every iteration and always has a model built using recent information.

This allows the creation of an on-line regression algorithm that exploits temporal structures (see Section 6.1), even when they are not known a priori with known performance penalties. We explore the topic and some of the pitfalls. For example; there

is a cost associated with allowing old experts to quickly become relevant again in the future with some interesting solutions in the literature, such as mixing past-posteriors by Bousquet et al [17].

We also continue the usage of specialist experts' evaluators and consider the effects of combining spatial dependencies with temporal evolution. We pick vicinities in the side information domain to create elementary time series based on homogeneous temporal structures derived from the strike domain. This time we use parsimonious time-series forecasters even though we are essentially working in a hybrid time-space domain. Prediction with expert advice bounds ensures that optimal vicinities are selected dynamically. The algorithm is also tested on the problem of predicting implied volatility of options and proves to be another viable alternative to sliding on-line regression and also validation-free.

Finally, we experiment with using specialist expert techniques to merge together ridge regression models that have randomly generated parameters.

## 1.2 Contributions and Organisation

These are the primary contributions of this thesis.

### 1.2.1 Music Segmentation Algorithm

An algorithm that finds a fixed number of homogeneous temporal structures in music data. The self-similarity of segments over a time horizon is computed avoiding some transient point-of-change heuristic pitfalls (see Chapter 3).

### 1.2.2 Merging Ridge Regression Models

Several new regression algorithms for on-line forecasting based on specialist expert techniques to exploit temporal structures in data. The algorithms are based on the following idea. A number of relatively small regions in the past are selected as train-

ing sets and instances of regression algorithms are trained on them. The resulting algorithms are then merged using specialist experts evaluators (see Section 4.5). The method thus constructed avoids both dealing with large matrices and retraining on the sliding window on every step of on-line kernel ridge regression. We apply the method to predicting implied volatility of options (see Section 6.2.1).

### 1.2.3 Merging Time Series Algorithm

An application of specialist experts' techniques to prediction with side information. We pick vicinities in the side information domain to create elementary time series based on temporal structures derived from the strike domain, use standard prediction techniques to predict for those elementary series, and then merge the predictions using specialist experts' methods. Prediction with expert advice bounds ensure that optimal vicinities are selected dynamically. The algorithm is tested on the problem of predicting implied volatility of options and proves to be a viable alternative to on-line regression (see Chapter 6.1).

## 1.3 Materials

The data and code for the experiments described in this thesis have been made available on-line.

- All the code for the music segmentation with the training set is available on GitHub<sup>2</sup>. The corpus ( $\approx 150\text{GB}$ ) we received from Denis Goncharov and Mikael Lindgren will be made available on request (it is in a cloud storage account and easily shareable).
- The code and corpus for the on-line fixed regions merging algorithm, the on-line lagged regions merging algorithm and the on-line variable window size merging

---

<sup>2</sup><http://github.com/ecsplendid/DanceMusicSegmentation>

algorithm are available on GitHub <sup>3</sup>.

- The code and corpus for the on-line merging time-series available on GitHub <sup>4</sup>.

## 1.4 Papers

- *A Long-Range Self-similarity Approach to Segmenting DJ Mixed Music Streams*. Springer collection Artificial Intelligence Applications and Innovations 2013 [109].
- *Segmentation of electronic dance music*. International Journal of Engineering Intelligent Systems for Electrical Engineering and Communications, 2014 [117].
- *Merging Time Series with Specialist Experts* <sup>5</sup>
- *Merging Specialist Region Experts* <sup>6</sup>

---

<sup>3</sup><http://github.com/ecsplendid/MergingRegionExpertEvaluators>

<sup>4</sup><http://github.com/ecsplendid/MergingTimeSeries>

<sup>5</sup><http://www.developer-x.com/papers/merging-time-series>

<sup>6</sup><https://www.developer-x.com/papers/merging-specialist-region-experts>

## Chapter 2

# Temporal Structures in Univariate Data

A temporal structure in the most abstract meaning is any physical or virtual structure in the corpora that has a time-dependency (immediate or long-term). This may manifest itself as a region structure.

### 2.1 Homogeneous Structures

We can infer a great deal about the temporal evolution of the underlying data from this the self-similarity image plot in Figure 2.1. Note that the distance function is symmetric so matrix has a copy of itself in the top-left half.

- White squares on the diagonal mean there is a region of *static-self similarity*. This means that structural changes in the data are not going to be observed in the immediate future.
- Thin white strips on the diagonal mean that the data is constantly evolving in respect of time (so it is not similar to itself in the past).
- Copies of diagonal structures away from the main diagonal means that the seg-

ment has been repeated in time (the distance away from the diagonal tells us how much time has elapsed before the repetition).

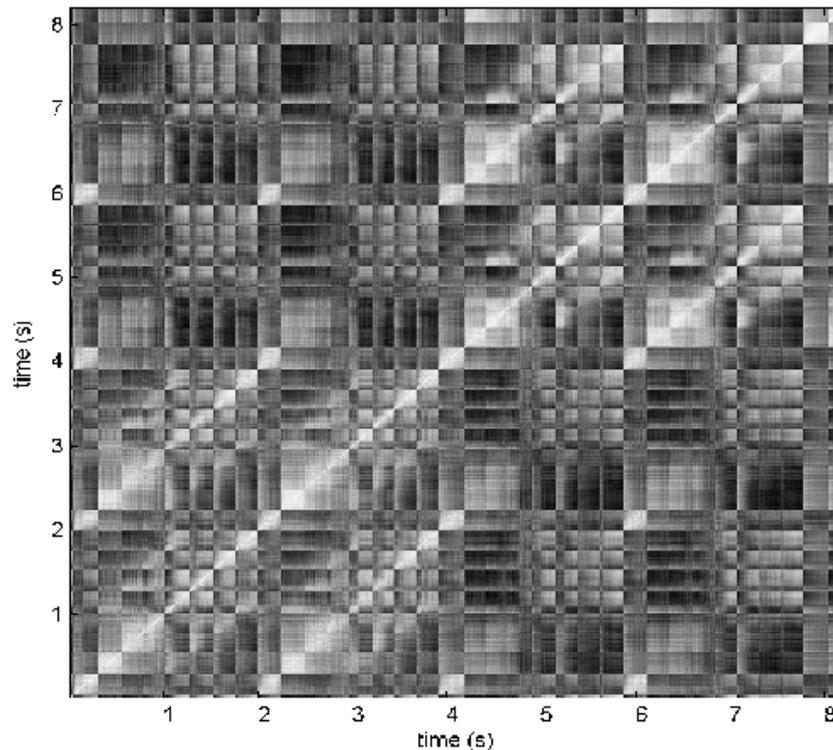


Figure 2.1: An example of a self-similarity matrix image plot (from Foote et al [48]) showing the first seconds of Bach's Prelude No. 1 in C Major, from The Well-Tempered Clavier, BWV 846. Whites indicate a high degree of similarity, and blacks – the opposite.

The most obvious example of a data structure is a region in a data set that exhibits the property of being self-similar (in-part or in-full).

Could segmentation or some other method to find these temporal structures in data sets lead to improved forecasting?

In Chapter 3, we first motivate an example by analysing a large corpus of music to find obvious temporal structures (songs). In Chapter 4 we use the same algorithm to see if similar temporal structures can be found on a univariate Russian Stock Exchange options futures volatility dataset which is highly interesting for on-line forecasting.

What was encouraging, was that we found some distinct temporal structures which were in some respects similar to the music data although more nebulous in form.

We developed an algorithm to exploit these temporal structures in an on-line setting (see Section 6.2.1), and then expanded the idea to work on temporal structures in the strike price domain (see Section 6.1).

## 2.2 Literature Review

In most scientific fields, metrics are instrumented sequentially over time. Observations of this type lead to an organisation of data called *time-series*. There are several tasks relevant to time-series data, these are chiefly; forecasting, classification and segmentation.

Models incorporating regime switching have a long tradition in empirical macroeconomics [40]. Key contributions include the work of Quant and Goldfeld dealing with the problem of discontinuous shifts in regression regimes at unknown points in the data series [56, 57, 105, 106].

A number of researchers have recently become interested in modelling economic and financial time series as subject to occasional, discrete shifts in parameters. Hamilton introduced in [60, 61] (and later applied by Chang-Jin [79]) an EM algorithm for obtaining maximum likelihood estimates of parameters for processes subject to discrete shifts (caused by temporal regimes) in autoregressive parameters with the shifts themselves modelled as the outcome of a discrete-valued Markov process. He later went onto introduce a marko-switching scheme for time-series detecting when new regimes were present [62].

As stated by Lovrić et al [88]; the main goal of segmentation is the extraction of time segments with similar features, or collectively dissimilar to the rest of the time period – the decomposition of time series into homogeneous segments having uniform characteristics (linearity, flatness, modality, monotonicity).

As summarised by Keogh et al [75, 76], many approached to time series in the literature use the so-called piecewise linear representation (PLC). The PLC is an approximation of a time-series which reduces it to a description of  $K$  straight lines. It is used for similarity searching [74], novel distance metrics, searching (finding motives) [31, 78, 87], clustering [77] and change point detection [53, 115].

Sclove segmented the U.S. gross national product time-series. The segments were considered as falling into classes. And a different probability distribution was associated with each class of segment. A Markov chain was used for a regime labels [111] and maximum likelihood estimation used for the generating the segmentation.

Appel et al [5] addressed the problem of adaptive segmentation of time series given abrupt changes in the spectral characteristics. The series were modelled as time series were modelled as zero mean gaussian distributed autoregressive (AR) processes

A typical application of time-series segmentation is in speaker diarization [118], in which an audio signal is partitioned into several pieces according to who is speaking at what times.

Audio diarization is the process of annotating an input audio channel with information that attributes temporal regions of signal to specific groupings (see Figure 2.2 for an illustration).

The three main domains for diarization are broadcast news audio, recorded meetings, and telephone conversations. The temporal boundaries are detected via the pathway or speech detection or change detection. Speech detection is often modelled as a classification problem [69, 89, 93, 113].

Change detection is finding points in the audio stream likely to be change points between audio sources. Typically distance metrics are used for this to see if adjacent windows in time are the same person speaking. The first approach as in [107] builds on the Bayesian information criterion [25] which locates change points inside a frame using a scored likelihood ratio test of whether the data in the frame suits a single (stay) or dual distribution (change point). Another method used here [114] and later [9, 52, 98]

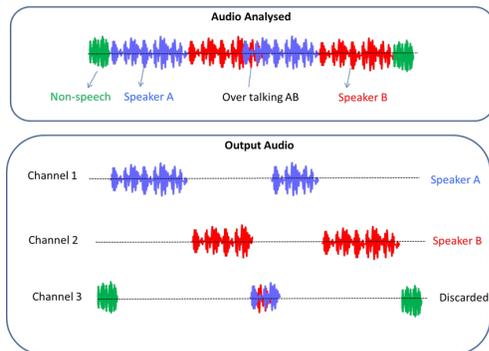


Figure 2.2: An illustration of speaker diarization.

represent discrete time intervals as fixed-length windows and models the KL-2 distance [66] between them.

Audio segmentation in the literature is colloquially implemented in the context of structural analysis. Music structure denotes the organization of a composition by its melody, harmony, timbre and rhythm. Repetitions, transformations and evolutions of music structure also contribute to its identity and it is this semantic information that structural analysis algorithms aim to extract from music. An example structure for a song might be *ABCABA*. Speaker diarization is another example of structural analysis.

Segmentation in the context of structural analysis has been thus far been concerned with creating a novelty function to find points-of-change using distance-based metrics, rather than trying to find a fixed number of segments in the most optimal setting. Heuristics with hard decision boundaries have been used to find the best change points, for example Tzanetakis [120] used first-order derivatives of a time series of audio features.

The use of a similarity matrix to visualize and analyse local time dependencies (at the time referred to as *recurrence plots*) was first proposed by Eckmann[42].

J. Foote [33, 44, 45, 46, 47, 49] was the first to use local self-similarity to spot musically significant changes in music. The distance or cosine angle between feature vectors can be used to construct a self-similarity matrix to visualise and exploit time dependencies in music data. The key assumption is that there is some kind of repetition

in the audio that can be spotted. The similarity matrix contains the distance between all feature vectors and a characteristic pattern develops where the diagonal elements are maximally self-similar and regions emerge representing segments of interest in the audio.

Footo correlated a Gaussian tapered checkerboard kernel[49] along the diagonal of a self-similarity (cosine) matrix derived from music features. This created a 1-dimensional novelty function that had the notion of self-similarity over a fixed time horizon. The kernel was ‘tapered’ down to zero towards the edges by a multiplicative Gaussian kernel to reduce edge noise. Our approach can be thought of as having a *soft* time horizon up to a fixed limit (Footo’s work had a fixed kernel size). However, the drawback of our method is that we find a fixed number of tracks. Naturally; the width of the kernel strongly determines the shape of the resulting novelty function. Small kernels will highlight transient changes in the audio while larger kernels will operate over a larger time horizon.

Goodwin et al. used a dynamic program for segmentation [59]. Their approach was to perform linear discriminant analysis to project features into an a priori learned feature space. Afterwards, Goodwin formulated the problem into one of finding the globally minimum cost path through a state graph (the so called ‘cluster space trajectory’) modelling local and transitional costs between segments. Goodwin already demonstrated in [58] that novelty peaks often exist within segments, not only on the boundary of segments and took the approach of modelling all possible sequential transitions between all possible segments.

A possible drawback to the approach by Goodwin and all other approaches in scene analysis segmentation that; is that they are somewhat local methods that focus on points-of-change rather than optimizing for the best possible results given a fixed number of segments to find. As Goodwin did not provide any results, we cannot conclude that a new distance function derived from learning a feature subspace improves segmentation accuracy.

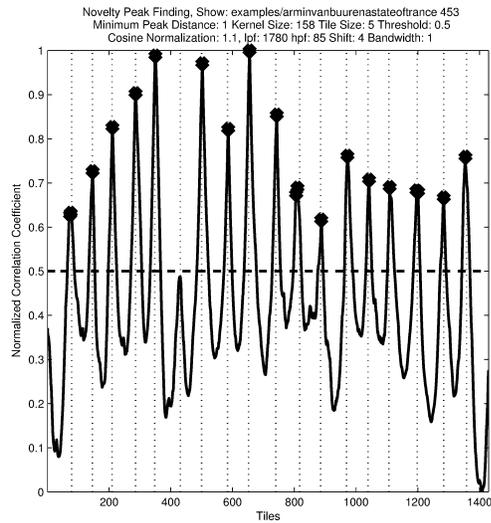


Figure 2.3: Foote’s novelty function for one of the radio shows in the corpus. The actual track indices are shown with dotted lines, and the predicted tracks are shown with the markers. Some of the parameters are drawn from our own method of constructing the self-similarity matrix (see Section 3.4.2)

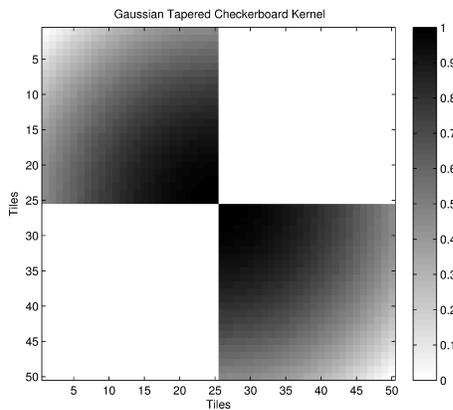


Figure 2.4: An example of Foote’s Gaussian tapered checkerboard kernel, width 50.

Clustering algorithms do exist that find a fixed number of segments but we have the added constraint that these segments need to be homogeneous, contiguous and time-dependent. Radu [37] demonstrated a time-dependent modification of agglomerative (hierarchical) clustering for segmentation of music. A constraint was added such that during the algorithm, clusters could only be merged if they were adjacent in time. Radu pre-processed the feature vectors to increase the homogeneity by averaging them over a sliding window of fixed size. Radu used a fixed number of clusters found as a stopping condition. The approach is model-free but by Radu’s own admission lacks any form of regularization, possibly producing clusters that vary in size significantly. Radu modelled the segmentation as a binary classification problem (later also used by Badawy et al [43]) subject to threshold time horizons allowing standard performance metrics such as precision, recall and  $F_1$  score to be used.

Stochastic model-based approaches to segmentation also have been devised. Levy et al use hidden Markov models in their approach to music segmentation in [85], building on their previous work [84, 86]. Their HMM is based upon a generative Gaussian mixture model where each segment has a set of states, each generating a Gaussian distribution of feature vectors. A musical audio file can then be regarded as having an underlying sequence of states that generates the observed feature sequence. The HMM can then be trained with a priori knowledge.

Plotz et al also used a an HMM for segmenting DJ mixed music streams, further developing the concept of generic acoustic generators first described in [11]. Unfortunately Plotz only evaluated his method against a small corpus of unknown origin (222 minutes with 65 song changes, compared to our 640 hours with 6757 song changes) and did not elaborate on the ground truth annotation methodology.

Badawy implemented a Foote inspired [49] segmentation scheme to segment 61 hours of recorded Montreux jazz festival concerts and compared to human captured ground truth meta-data.

## Chapter 3

# Discovering Temporal Structures In Music

In this chapter, the problem of finding temporal structures in DJ-mixed dance music recordings (pod-casts, radio shows, live events) is considered. In this domain setting; a temporal regime structure is synonymous with a homogeneous music track. After some discussion; an algorithm is presented to find these tracks as close as possible to what a human domain expert would find in respect of the same task given a fixed number of songs to find. The algorithm is optimized for the scenario when the number of tracks is known a priori although is also capable of estimating the number of tracks and is evaluated in both circumstances. As the number of segments is known in advance; reliance on local points-of-change heuristics prevalent in common segmentation algorithms is unnecessary.

The goal of DJ-mixing is to render track boundaries effectively invisible from human perception. Segmentation is performed on a self-similarity matrix which is derived from normalized cosines of various cost matrices which have themselves been derived from a time-series of Fourier based spectral features. The cost matrices proposed here introduce notions of general self-similarity and also specific notions such as; symmetry, contiguity and evolution in respect of time. The segmentation configuration is

parametrized and an evolutionary algorithm is executed on a small test set to find optimal parameters for the task of segmentation.

Our algorithm is quantitatively assessed on a large corpus (640 hours) of radio show recordings which have been hand-labelled by a domain expert. The method presented could be used on other corpora to discover temporal structures.

### 3.1 Electronic Dance Music

Electronic Dance Music tracks are usually mixed by a disc jockey (DJ). For this reason EDM music streams are unique compared to other genres of music. Mixing is the *modus operandi* in electronic music. First, the audio file is transformed into a time series of features and discretized into adjacent tiles and transformed into a domain where most pairs from the same track would be distinguishable by their cosine.

*Contiguous-segmentation* differs from the standard clustering problem in that the clusters arrive sequentially and are contiguous (AAABBBCCDDDD, not AAABBBCCBBBB). This may also be known as *time-dependent* clustering and is related to *homogeneous* clustering. For brevity the term *segmentation* will be used from now on to describe this configuration. The intuition behind the word homogeneous is that segments that have intra-segment similarity and inter-segment dissimilarity are desirable.

Music and mixes of music have the property that they are made up of recursively repeating self-similar regions within segments. Our method does not strictly require any training or tenuous heuristics to perform well. The distinguishing feature of our problem domain is that the number of segments is known a priori but the segmentation boundaries are not known, or ambiguous and subjective. However, computing the best solution is desirable.

Our features are based on a Fourier transformation with convolution filtering to accentuate prominent instruments and therefore self-similarity within tracks. A similarity matrix is created from these cosines and then cost matrices derived showing the costs

of fitting a track at a given time with a given length. Dynamic programming is used to create the cost matrices and again to perform the most economical segmentation of the cost matrices to fit a fixed and predetermined number of tracks. The number of tracks can be estimated using the same framework. Dynamic programming means solutions to a problem are described in terms of overlapping sub-solutions to achieve a significant improvement in time complexity and therefore execution time.

The intended purpose of the algorithm is to reconstruct globally optimal boundaries given a fixed number of tracks known a priori. The self-similarity of segments over a time horizon is scrutinised avoiding some transient point-of-change heuristic pitfalls. The track listing is usually published by the DJ which is why the number of tracks is known. The use case is when one has recorded a show (perhaps automatically), downloaded a track list and needs to reconstruct the indices given that track list. The order of the reconstructed indices is critical so that the track names with the appropriate indices can be aligned.

One of the interesting features of audio is that you cannot scrub through it, and get an overview in the same way you can with video. Audio has a reduced continuum of context when one scrubs through it. Perhaps due to the lack of redundant, persistent scene-setting information or indeed a psychological reason. Even in video applications, discovery, context and scrubbing are an active area of research [94]. Time meta-data would allow click through monetisation, and allow improved use-case scenarios. For example; publishing track names to social networks, information discovery and retrieval. Capturing meta-data in audio is a time consuming and error-prone process. Tzanetakis [120] found that it took users on average 2 hours to segment 10 minutes of audio using standard tools. While not directly relevant one might glean from those findings that there is a strong motivation to automate this process.

DJs always match the speed or beats per minute (BPM) of each adjacent track during a transition and align the major percussive elements in the time domain. This is the central concept of removing any cognitive dissonance from playing overlapping

tracks. Tracks can overlap by any amount. DJs increase adjacent track compatibility further by selecting pairs that are harmonically compatible (aligned and congruent in the frequency domain) and by applying spectral transformations; such as equalizer filters (EQ).

The dance music sub-culture has grown significantly over the last 20 years and music mixing has become an art-form. High quality music streams of DJ mixed music are increasingly ubiquitous.

See Figure 3.3 for an annotated spectrogram of a segment of electronic dance music.

## 3.2 Music Corpora

We have been supplied with several broadcasts from three popular radio shows. See Table 3.2 for a description. The show genres are a mix of so called ‘progressive trance’, ‘uplifting trance’ and ‘tech-trance’.

There are no silent gaps in the recordings. The shows come in 44100 samples per second, 16 bit stereo MP3 files sampled at 192Kbs. The shows were re-sampled to 4000Hz 16 bit mono (left+right channel) WAV files to allow us to process them faster. The *Sound eXchange*<sup>1</sup> program was used to do this. These shows are all 2 hours long. The overall average track length is 5 and a half minutes (slightly less for Magic Island, see Figure 3.2) and normally distributed.

An additional dataset of 36 radio shows have been mixed by and annotated by Mikael Lindgren (the so called `lindmik` dataset). These shows are extremely useful because the DJ is the same person who created the ground truth time annotations which should in theory reduce the amount of human confusion present in his annotations. Also there is less noise, for example voice-overs, guest mixes, radio show sounds, introductions etc. These shows also vary significantly in length from 1 hour to nearly 5 hours. There are 339 shows in total.

The corpora is believed to be the largest of its kind used in the literature going on

---

<sup>1</sup><http://sox.sourceforge.net>

Table 3.1: Descriptive statistics about the corpus.

REFERENCE	NAME	DJ	HOURS	MEAN TRACKS	SUM TRACKS	SHOWS	TRK. LENGTH (S)
ASOT	A State of Trance	Armin van Buuren	198	20.6	2247	109	317.9
MAGIC	Magic Island	Roger Shah	198	17.3	1839	106	388.2
TATW	Trance Around The World	Above & Beyond	162	20.1	1771	88	329.8
LINDMIK	On Cue	Mikael Lindgren	83	25	900	36	331.1
			641	20.7	6757	339	341.7

the comparative table of segmentation corpora listed by Peiszer et al in their literature review of audio segmentation [101]. More recently Badawy et al [43] used a corpus of 61 hours. The corpora is longer than 640 hours in length.

There is already a large community of people interested in getting time annotations for DJ sets. *CueNation*<sup>2</sup> is an example of this. CueNation is a website allowing people to submit *cue-sheets* for popular DJ mixes and radio shows. A cue-sheet is a text file containing time annotation meta-data (indices) for a media file.

The three main radio shows in the corpus were hand captured by Denis Goncharov; a domain expert and one of the principal contributors to *CueNation*. One of the significant problems with this task is that there is a small but apparent amount of *confusion* associated with the human captured indices (see Section 3.3 for details) where 5% of tracks get placed on a different musical *bar* (see following quote for description). On some tracks, it is unclear where to place the optimal index on the macro scale (30+ seconds rather than frames) and when analysing the human annotations, somewhat obvious human errors are apparent. Many of the cue-sheet authors themselves reject the idea of automating the task, citing the poor precision of any such result (they often place indices on the exact MP3 frame). However this sentiment seems misplaced given that they frequently make mistakes or that it is a matter of opinion where to place the track and some consistent method may be preferential. A potential outcome of our method could be an assistance mechanism to help with initial placements. Our results demonstrate that it is indeed possible to automate this task and that while there is some uncertainty attached to the optimal placement, it is still predictable. Indeed, the

<sup>2</sup><http://www.cuenation.com>

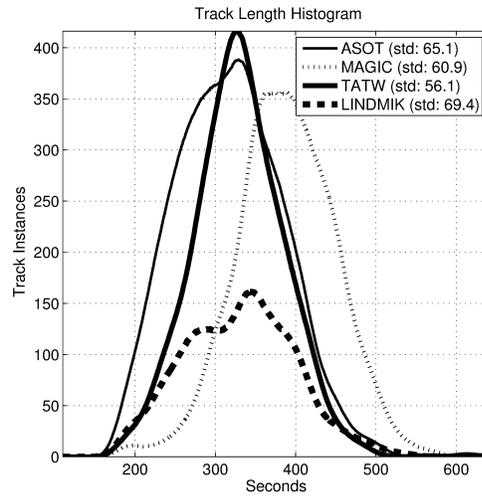


Figure 3.1: Track length histogram for all shows in the corpus.

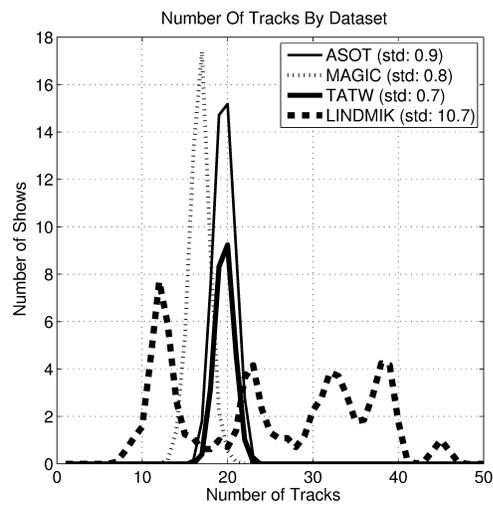


Figure 3.2: Number of tracks in each show for each dataset. The lindmik dataset is highly variable.

uncertainty is not small on the majority of track indices.

Denis Goncharov provided us with the following description of how he captures the indices. To quote from a personal email exchange with Denis:

Trance music is made in slices of 8 bars. 1 bar is 4 beats. At 135 beats per minute, 8 bars is  $(60 / 135) \times 4 \times 8 = 14.8$  sec. Trance music tends to be around 130-135 BPM. It is a matter of personal preference which point of the transition to call the index. My preference is to consider the index to be the point at which the second track becomes the focus of attention and the first track is sent to the background. Most of the time the index is the point at which the bass line (400Hz and lower) of the previous track is cut and the bass line of the second track is introduced. If the DJ decides to exchange the adjacent tracks gradually over the time instead of mixing them abruptly then it is up to the cue-sheet maker to listen further into the second track noting the musical qualities of both tracks and then go back and choose at which point the second track actually becomes the focus of attention.

The most obvious and pervasive element in dance music is the percussion (the beats). It appears that ignoring the percussive information is advantageous, because DJs use percussion primarily to blur boundaries between tracks. Experimentation with capturing percussive based features proved unsuccessful because transitions between tracks and groups of tracks appeared as stronger self-similar regions than the actual tracks. The percussive feature extractor transformed the autocorrelation of the audio samples in the time domain tiles, and compared the cosine of their absolute values. It was reasonably clear from the research that track boundaries are revealed with greater clarity between changing harmonically self-similar regions (instruments). However. Percussive features might be an interesting research direction in the future because useful information is currently being ignored.

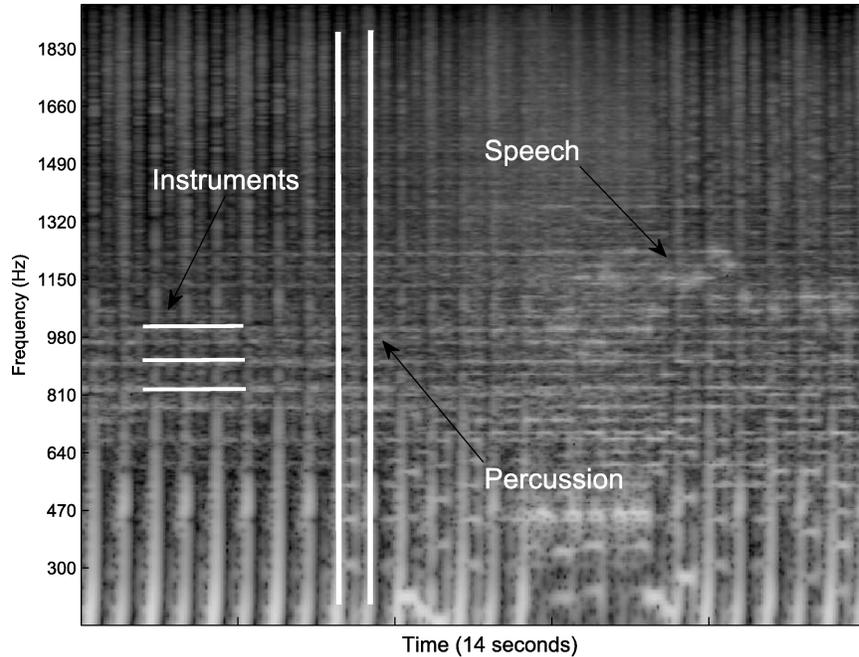


Figure 3.3: Illustration of dance music in a spectrogram with features marked. Percussive elements have full spectrum activity, while harmonic instruments look like horizontal train tracks.

Some DJs mix *harmonically* (by matching instruments instead of percussion) but this preys on human hearing and perception. For an instrument to sound harmonious to a human; it needs to satisfy the constraint of not stimulating any single *critical band* ([16]) of hearing more than once ([103]). An algorithm capturing the harmonic information would still be able to distinguish two harmonically compatible tracks because they may still be different in the spectrum. For more information on harmonics see [8, 12, 119].

### 3.3 Human Accuracy

Some quantitative analysis was performed on how accurate the humans themselves are at creating indices. In the absence of a perfect data set our analysis hinged on the amount to which the humans disagreed with each other, aggregated over a large

amount of historical data. Mikael Lindgren was kind enough to send us a dump of his cuesheet database to experiment with. As ASOT is such a popular show there were many independently captured cuesheets to compare against for all of the historical shows. All shows were selected having at least 3 distinct cuesheets (not exact copies or shifted/misaligned copies of each other) and such that all the cuesheets had the same number of tracks. The first track was ignored (as it was always 0 seconds). The result was 115 shows with 3 authors. 65 shows with 4 authors and 30 shows with 5 authors. A histogram was generated of distances from the median time for each track, for each cuesheet and assumed values greater than 100 seconds or less than  $-100$  seconds were outliers. The standard deviation of the *human disagreement* variable is 9.13 seconds. See Figure 3.4 for an illustration. So at this stage it does not seem feasible to achieve a higher accuracy when evaluating against a method which is intrinsically error prone. An important caveat here is that ASOT turned out to be the most error-prone show to segment out of our corpus. The standard deviation of the bumps could be reduced if times were normalized by the BPM of each transition. The *bar-scale* confusion peaks centered around  $\pm 14.8$  seconds, and  $\pm 29.6$  seconds represented 5 percent of the total annotations.

## 3.4 Data Handling

### 3.4.1 Preprocessing

The corpus had some outliers that may have slightly distorted the analysis of our method. Many of the tracks in the ground truth annotations for our corpus were actually introductions or voice-overs. Almost all of these outlier tracks were short in length. To ameliorate the situation any tracks that were shorter than 180 seconds were removed (which are clearly not normal according to Figure 3.2). The same went for any end tracks that were shorter than 240 seconds as very often the end tracks on a radio show contain peculiar elements (for example voice-overs, interviews, show-

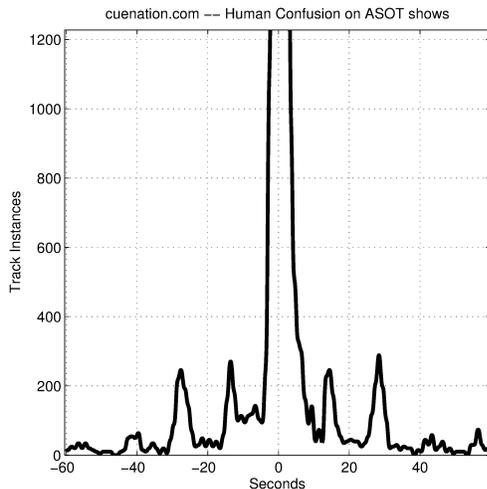


Figure 3.4: Illustration of the ‘human disagreement’ random variable (zoomed in at the bottom, the peak at 0 seconds is 4708 tracks), standard deviation 9.13 seconds. Peaks are visible at intervals of 8 bars (14.8 seconds) which corroborates the analysis from Denis Goncharov in Section 3.2. The 4 adjacent error clusters account for roughly 5 percent of the total number of tracks. The variance around the peaks represents the BPM variance in *asot*.

related ‘*jingles*’). This required some manipulation of the cue-sheets and audio files. The undesirable segments of the audio files were chopped out, and the cue-sheets were re-flowed so that the time indices point to the correct location in the file.

The algorithm still performs similarly when removing just these indices and leaving the audio intact underneath, so it would not significantly affect any real-world implementation.

For those wishing to use this algorithm in practice with pre-recorded shows; the introductions at the start of the shows are often fixed length or at least predictable so error would be small on average.

The `lindmik` dataset which was noise-free did not require any preprocessing whatsoever.

### 3.4.2 Feature Extraction

#### 3.4.2.1 Music

*Sound eXchange* was used (see Section 3.2) to downsample the shows to 4000Hz. Frequencies above around and above 2000Hz were not interesting because instrument harmonics become less visible in the spectrum as the frequency increases. The Nyquist theorem [100] states that the highest representable frequency is half the sampling rate, so this explains our reason to use 4000Hz. Let  $R$  denote sample rate. Let  $L$  be the length of the show in samples.

Fourier analysis facilitates the representation of a time domain process as a set of integer oscillations of trigonometric functions. The tiles are transformed into the frequency domain using the discrete Fourier transform

$$F(x_k) = X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi \frac{k}{N}n}$$

which transforms a sequence of complex numbers  $x_0, \dots, x_N$  into another sequence of complex numbers  $X_0, \dots, X_N$  where

$$e^{-i2\pi \frac{k}{N}n}$$

are points on the complex unit circle. Note that the fftw algorithm [51] that is used to perform this computation operates significantly faster when  $N$  is a power of 2 so the input is zero padded to the next power of 2. Let  $M$  denote the tile width in seconds (an algorithm parameter). Note that

$$N = \frac{L}{M}$$

denotes the tile size in samples (length of show in samples over the tile size). Let

$$T = \left\lfloor \frac{L}{\tilde{M}} \right\rfloor$$

be the total number of tiles, and

$$\tilde{M} = \frac{L}{N}$$

the tile width in samples. Because real values are passed into the  $F(x_k)$ , the second half of the result is a rotational copy of the first half.

Show samples are collated into a time series  $Q_i^y$  ( $T \times N$ ) of contiguous, non-overlapping, adjacent *tiles* of equal size where  $i = 1, 2, \dots, T$ . Samples at the end of the show that do not fill a complete tile get discarded. The affect of this is increasingly negligible with decreasing tile size. Since  $N$  is zero-padded to the next power of two, this also decreases the affect.

The entire range of the spectrum is not always interesting, so let  $l$  represent a low pass filter (in Hz) and  $h$  the high pass filter (in Hz). So the range from  $h$  to  $l$  is captured on the first half of the result of  $F$ . Let  $\hat{h} = \lceil h \cdot \frac{N}{R} \rceil + 1$  be the position of  $h$  in the spectrum, and  $\hat{l} = \lceil l \cdot \frac{N}{R} \rceil + 1$  be the position of  $l$  in the spectrum.

Let  $D_e^y$  ( $T \times \hat{l} - \hat{h} + 1$ ) denote the feature matrix.

For each tile  $\bar{i} = 1, 2, \dots, T$  assign

$$D_{\bar{i}}^{1, \dots, \hat{l} - \hat{h} + 1} = \left| F(Q_{\bar{i}}^{1, \dots, \tilde{M}})_{\hat{h}, \hat{h} + 1, \dots, \hat{l}} \right|$$

selecting the part of the spectrum between the high and low pass filters  $h$  and  $f$ . Take the absolute values of the complex result of  $F(x_k)$  (defined as its distance in the complex plane from the origin using the Pythagorean theorem).

To accentuate instrument harmonics convolution filtering is performed on the feature vectors in  $D$ , using a Gaussian first derivative filter. This works like an edge detection/transient filter (see [10, 72, 92, 95]) but also expands the width of the tran-

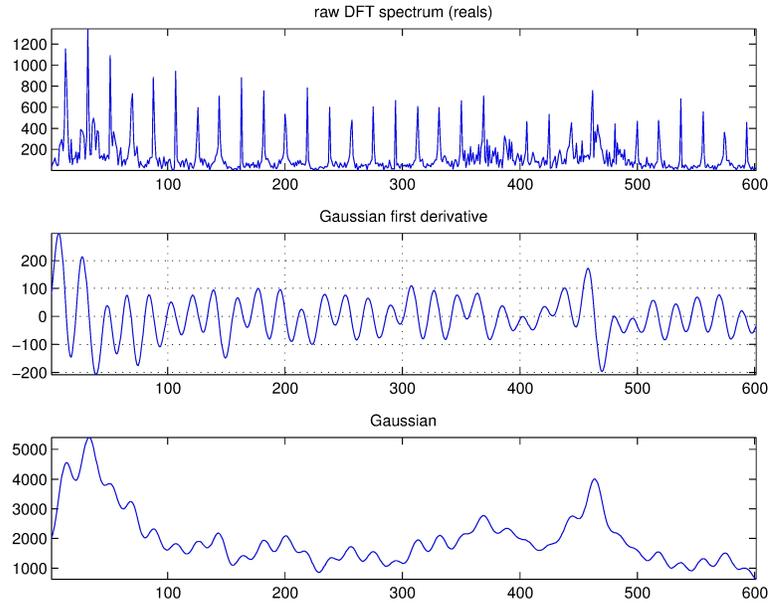


Figure 3.5: An illustration of convolution filtering applied to the top signal; the Gaussian first-derivative, then the standard Gaussian below that. Note that both filters widen/soften the peaks but the Gaussian first-derivative centres around zero when no extreme transient is present.

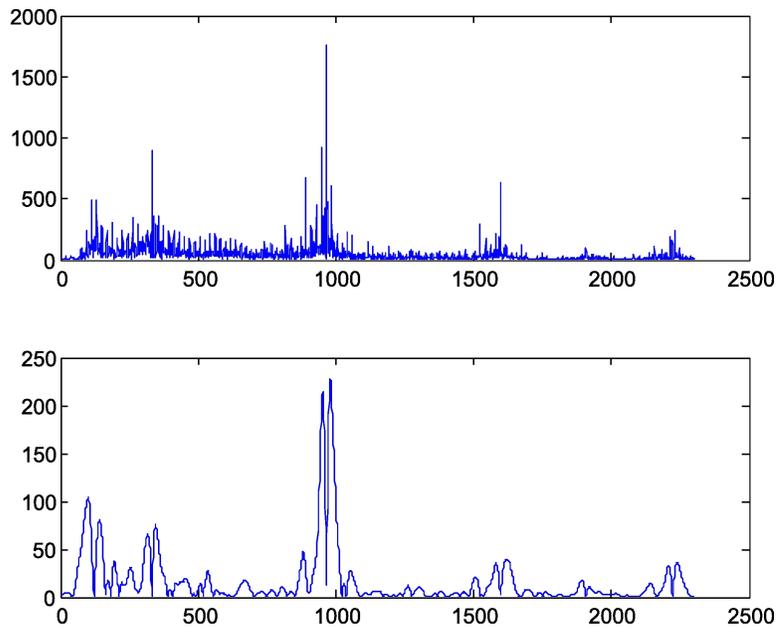


Figure 3.6: An example of the filtering that is applied to an input signal.

sients (instrument harmonics) to ensure that feature vectors from the same song appear similar because their harmonics are aligned on any distance measure (cosines are used). This is an issue because of the extremely high frequency resolution from having such large inputs into  $F(t_i)$ . For example with a tile size of 10 seconds and a sample rate of 4000; the frequency resolution is  $0.5 \cdot 10 \cdot 4000 = 20\text{KHz}$ . See Figure 3.6 for an illustration.

Typically a ‘short-time discrete Fourier transform’ is used which has smaller sized inputs (windows) into  $F(t_i)$  which are usually overlapping and are multiplied by a window function, attenuating the tails to reduce spectral leakage. Usually these window functions look similar to a Gaussian, for example;

$$\text{Hann}_i = 0.5 - 0.5 \cos \frac{2\pi i}{n-1} w(i)$$

where  $n$  is the window size (see [121] for an example). The short-time Fourier transform is relevant when increased time precision is needed as there is a frequency-time resolution trade-off with respect of the input size to  $F(t_i)$ . This is not a concern in this particular application as our time resolution is never required to be better than 1 second which would still produce adequate frequency resolution.

The Gaussian first derivative filter is defined as

$$-\frac{2\hat{\lambda}}{v^2} e^{-\frac{\hat{\lambda}^2}{v^2}}$$

where

$$\hat{\lambda} = \{ -\lfloor 2v \rfloor, \lfloor -2v + 1 \rfloor, \dots, \lfloor 2v \rfloor \},$$

and

$$v = b \frac{N}{R}.$$

$b$  is the bandwidth of the filter in Hz and this is a parameter of the algorithm. After

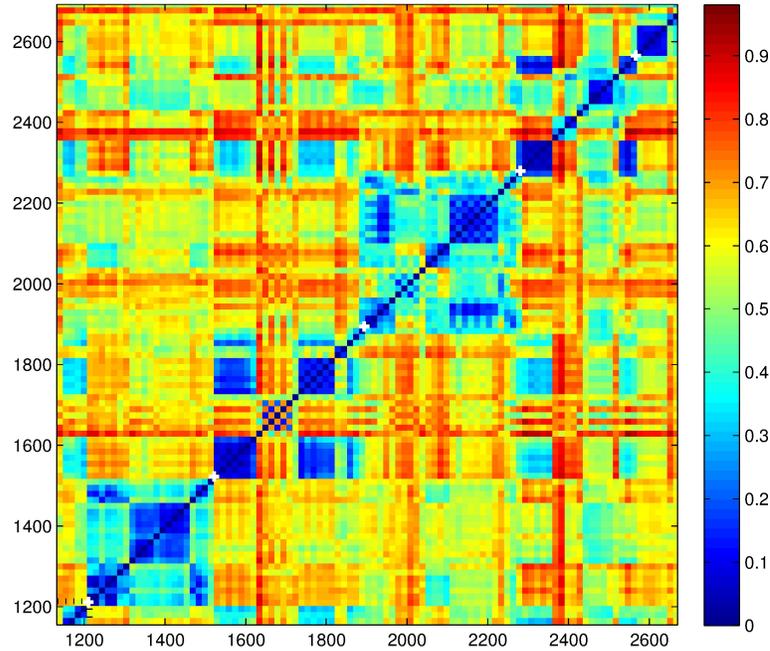


Figure 3.7: Visualization of  $S$ . The tracks are marked with white crosses. This is a quite an easy example (the tracks are clearly distinguishable).

the convolution filter is applied to each feature vector in  $D$ , absolute values are taken and normalized on the vector lengths.

Because the application domain is well defined in this setting, features can be designed that look specifically for interesting elements (musical instruments). Typically in the literature; algorithms use an amalgam of general purpose feature extractors. For example; spectral centroid, spectral moments, pitch, harmonicity [120]. A dissimilarity matrix of cosines is constructed as is common in the literature for similar applications [45]. The cosines are computable easily because they are the the inner products of the respective features (the features have been normalized to unit length).

### 3.4.2.2 Self-Similarity Matrix

Let

$$S_{ij} = 1 - \langle D_i, D_j \rangle,$$

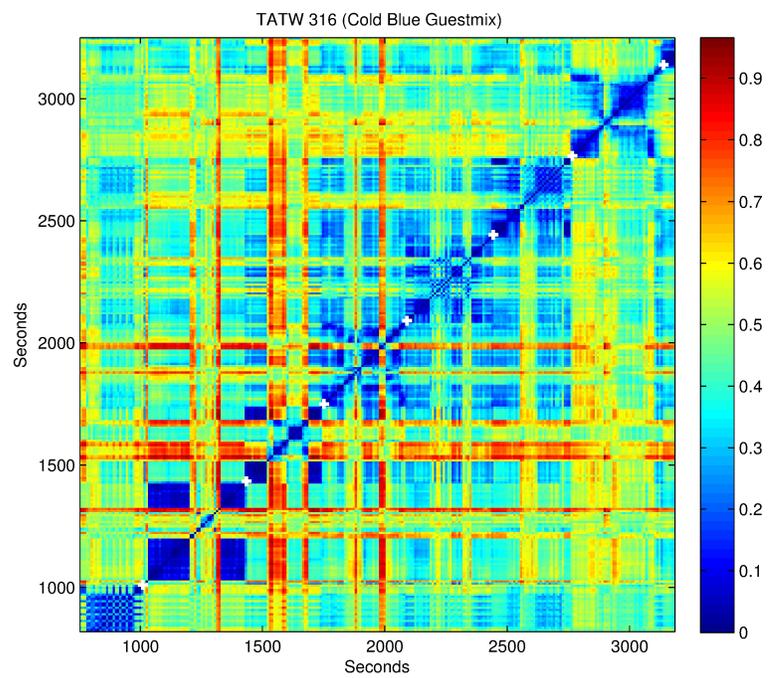


Figure 3.8:

Figure 3.9: Visualization of  $S$ . The tracks are marked with white crosses. This is a hard example as it is ambiguous.

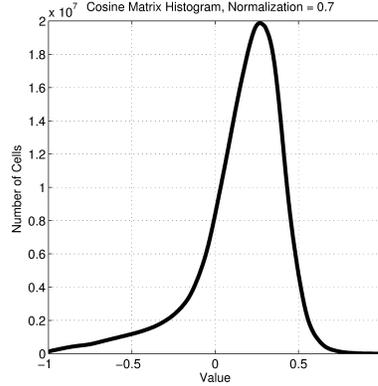


Figure 3.10: Illustration of the effect of normalization parameter  $\hat{c} = 0.7$  on the values in  $S$  on radio show `asot` 453. The small raised section on the left correspond to the tracks down the diagonal.

define the dissimilarity matrix of cosines.

Next, normalizing transformations are applied. First center  $S$  around 0.5 by raising each element to the power  $2s$ , where  $s = \frac{1}{T^2} \sum_{i,j=1}^T S_{ij}$ . Since for  $x \in [0, 1]$  and  $y > 0$ ,  $x^y \leq x$  if  $y \geq 1$  and  $x^y \geq x$  if  $y \leq 1$ , the transformation  $S_{ij} \rightarrow S_{ij}^{2s}$  increases the values  $S_{ij}$  whenever the mean value  $s < 0.5$  and decreases them whenever  $s > 0.5$ . Note that the transformation keeps the values  $S_{ij}$  in the interval  $[0, 1]$ . This a convenient and gentle way to rescale  $S$ .

Secondly raise each value  $S_{ij}$  to a power  $\hat{c} \in [0.5, 1.5]$  and then rescale and translate them to  $[0, 1]$  using  $S_{i,j} \rightarrow 2S_{i,j} - 1$ . The parameter  $\hat{c}$  is tuned so as to achieve the right balance between negative *incentives* and positive *disincentives* for meaningful track placement. As reported in [109] there is a pitfall of self-dissimilar regions within tracks negatively affecting the cost of placement. The distribution of values in  $S$  after the transformations will have a raised tail on left. This will become relevant when cost matrices are discussed as some of them depend on the sign of the value in  $S$ .

See Figure 3.11, 3.7 and 3.9 for a visualization of  $S$  and Figure 3.10 for an illustration of the normalization.

### 3.4.2.3 Cost Matrices

Let  $w$  and  $W$  denote the minimum and maximum track length in seconds, these are be parameters of the algorithm that will help improve the time complexity.

The cost matrix  $C(f, t)$  is constructed that describes the cost of placing a track starting at  $f$  and finishing at  $t$  (and having length  $t - f + 1$ ). After making some observations in  $S_{ij}$ , cost matrices have been created that exploit observed phenomenological temporal structures. Our suspicion is that believe this phenomena is not intrinsic to dance music and is indeed prevalent in nature.

An additional cost matrix which is just a 1-dimensional Gaussian random function centred around the mean track length for all times which can be used to regularize the other matrices or used on its own as a comparator to a more *naïve* method of placement.

The cost matrices described in this section exploit themes such as contiguity, symmetry and evolution as well as simple summation of  $S$  as reported in [109]. In [109],  $S$  was on the interval  $[0, 1]$  and the summation method could only consider disincentives. The new cost matrices have a parameter to shift the consideration of incentive versus disincentive and values on the interval  $[-1, 1]$ .

On the whole, a significant number of tile pairs within one track are similar to each other. Pairs of tiles that do not belong to the same track are expected to be dissimilar, most of the time. However, tracks have contiguous regions within them that are dissimilar to each other. Transitions between songs may appear as a self-similar region but usually also similar to each adjacent track to varying degrees.

**3.4.2.3.1 Summation** The most obvious strategy of all is to sum up all relevant tiles in  $S$  for each candidate track from tile  $f$  through tile  $t$ . Let  $C(f, t)$  define the cost of a candidate track from tile  $f$  through tile  $t$ , to be the sum of the similarities between all pairs of tiles inside it

$$C(f, t, \omega, \bar{S}) = \sum_{i,j=f}^t \frac{\bar{S}}{(t-f+1)^\omega}$$

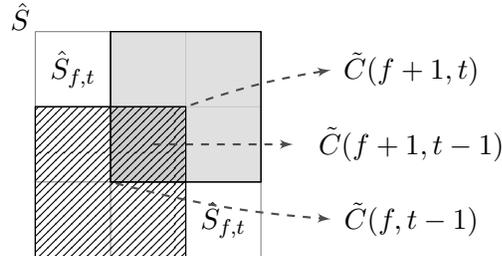
where

$$\bar{S} \leftarrow \hat{S}_{ij}(\Omega) = \begin{cases} \Omega S_{ij}, & \text{if } S_{ij} > 0 \\ (1 - \Omega) S_{ij}, & \text{otherwise} \end{cases}$$

for all  $i, j \in S$ . On this cost matrix,  $\bar{S}$  and  $\Omega S_{ij}$  are the same. Later this summation function for modified  $\bar{S}$  will be used. The quantity  $\hat{S}$  is a modification of  $S$  incorporating the *incentive bias* parameter  $\Omega$  which controls the balance of positive and negative values. Direct computation using the definition takes  $O(TW^3)$  time.  $O(TW)$  can be improved by using the following recursion for the unnormalized quantity  $\tilde{C}$  (assume that  $f+1 \leq t-1$ , and note that the incentive bias parameter  $\Omega$  has been temporarily removed for clarity):

$$\tilde{C}(f, t) = \tilde{C}(f+1, t) + \tilde{C}(f, t-1) - \tilde{C}(f+1, t-1) + \hat{S}_{ft} + \hat{S}_{tf}.$$

The recursion implies that the cost of a track of length  $L = t - f + 1$  can be calculated from the costs of shorter tracks using a constant number of operations. The following picture provides an illustration in the domain of  $\hat{S}$ :



It is useful to scale cost matrices onto the interval  $[0, 1]$ . Let normalization function

$$N(x) = \frac{x - \min_{ft} x(f, t)}{\max_{ft} x(f, t) - \min_{ft} x(f, t)}$$

and

$$\hat{N}(x) = (2N(x)) - 1$$

This normalization is then applied; Let  $C \leftarrow \hat{N}(C)$ .

See Figures 3.12 and 3.13 for an image visualization of the summation cost matrix with different incentive biases.

**3.4.2.3.2 Symmetry** A common feature on dance music tracks is partial mirror-symmetry. A cost matrix is built to capture that; as follows.

Let  $\Lambda(f, t, d)$  be the diagonal parallel to the minor diagonal of  $S$  and at the ‘distance’  $d$  from it. It is represented as an ordered set

$$\Lambda(f, t, d) = \langle S_{f+d,f}, S_{f+d+1,f+1}, S_{f+d+2,f+2}, \dots, S_{t,t-d} \rangle.$$

For each such diagonal in one triangle/half of  $S$  each element is compared against its mirror counterpart. For an ordered set  $\Lambda$  its cost is defined as

$$\bar{C}(f, t, \bar{\Omega})(\Lambda, \bar{\omega}) = \sum_{i=1}^{|\Lambda|} \frac{\delta(\Lambda_i, \Lambda_{|\Lambda|-i+1}, \bar{\Omega})}{i^{\bar{\omega}}}$$

where

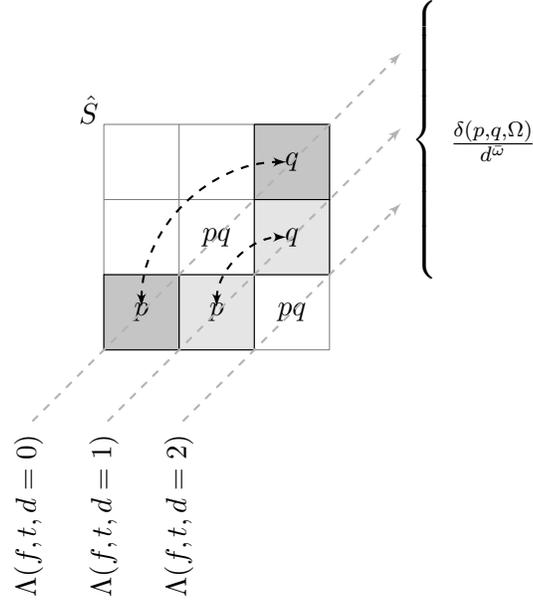
$$\delta(p, q, \Omega) = \begin{cases} 0, & \text{if } \text{sign}(p) \neq \text{sign}(q), \\ \Omega pq, & \text{if } \text{sign}(p) \geq 0 \text{ and } \text{sign}(q) \geq 0, \\ (1 - \Omega) pq, & \text{if } \text{sign}(p) < 0 \text{ and } \text{sign}(q) < 0, \end{cases}$$

i.e., ‘symmetric’ pairs that have the same sign make positive contributions and pairs that have a different sign contribute 0 to the cost. Let cost matrix

$$\bar{C}(f, t, \bar{\Omega}, \bar{\omega}) = \sum_{d=1}^{t-f+1} \bar{C}(\Lambda(f, t, d, \bar{\Omega}), \bar{\omega})$$

Finally, let  $\bar{C} \leftarrow \hat{N}(\bar{C})$ .

Clearly, one can reuse the cost for shorter intervals to calculate the cost of longer ones, namely,  $\bar{C}_{f+1,t-1}$  can be used to calculate  $\bar{C}_{ft}$  this saving computation time. See implementation on *GitHub* (see Section 1.3) with a time complexity of  $O(TW^2)$



**3.4.2.3.3 Static Contiguity** Horizontal contiguous traces in  $\hat{S}$  indicate that the track is self-similar (negative values) or self-dissimilar (positive values) due to repetition. If a given tile is the same as a set of contiguous tiles following it, then there is some static contiguous region in the show. The word *static* denotes that the music is not evolving in respect of time (which would instead create a diagonal trace in  $S$ ).

The approach of modifying  $S_{ij}$  in place is employed (See Algorithm 2) so that the fast summation algorithm described in Section 3.4.2.3.1 can be used. The algorithm takes the  $n$ th order differences (note that  $\text{diff}(x, 2) = \text{diff}(\text{diff}(x))$ ) in two dimensions of  $S_{ij}$ , past and future. For simplicity our version of the  $\text{diff}$  function will return a matrix of equal size, zero-padded at the beginning.

Note that this algorithm introduces new parameters:  $\dot{p}$  for the past  $n$ th order differences,  $\dot{f}$  the future  $n$ th order differences,  $\overleftarrow{\Omega}$  the past differences incentive bias,  $\overrightarrow{\Omega}$  the future differences incentive bias.  $\bar{p}$ , how much contribution the past differences

make, and  $\bar{f}$  the future differences contribution.  $\bar{n}$  is the normalization coefficient for this cost matrix.

---

**Algorithm 1:** Construct *contig-static* dissimilarity matrix by modifying  $S_{ij}$  in-place.

---

- 1  $\dot{s}_{ij} = \text{sign}(S_{ij})$
  - 2  $P_{ij} \leftarrow \text{diff}(\hat{S}(\overleftarrow{\Omega})^\top, \dot{p})^\top$ ; ( $\dot{p}$ th order differences, zero-padded at the start by  $\dot{p}$  so the size of  $F$  remains unchanged)
  - 3  $P_{ij} \leftarrow N(P)$
  - 4  $P_{ij} \leftarrow \bar{p}P_{ij}$ , for all  $i, j \in P$
  - 5  $F_{ij} \leftarrow \text{diff}(\hat{S}(\overrightarrow{\Omega}), \dot{f})$ ; ( $\dot{f}$ th order differences, zero-padded at the start by  $\dot{f}$  so the size of  $F$  remains unchanged)
  - 6  $F_{ij} \leftarrow N(F)$
  - 7  $F_{ij} \leftarrow \bar{f}F_{ij}$ , for all  $i, j \in F$
  - 8  $\dot{S} \leftarrow \dot{s}|P + F|$
  - 9  $\dot{S} \leftarrow \hat{N}(\dot{S})$
  - 10  $\text{diag}(\dot{S}, d) \leftarrow \text{diag}(\dot{S}, d)d^{\bar{n}}$  for all  $d \in \{1, 2, \dots, W\}$ ; (multiply all elements in diagonals  $d$  of  $\dot{S}$  by  $d^{\bar{n}}$ )
  - 11  $\dot{S} \leftarrow \hat{N}(\dot{S})$
  - 12 **return**  $\dot{S}$
- 

$\dot{S}$  is then transformed into a cost matrix  $\vec{C}(f, t, \bar{n}, \dot{s}, \dot{p}, \dot{f}) = C(f, t, \omega, \dot{S})$  using the summation function described in Section 3.4.2.3.1.

**3.4.2.3.4 Evolutionary Contiguity** Any diagonal traces in  $S$  that are parallel to the main diagonal are partial copies of the track in the future which evolve in respect of time (self-similar tiles or groups of tiles that are dissimilar to the previous or following tiles). Evolutionary contiguity is a diagonal version of the static contiguity cost matrix described in Section 3.4.2.3.3.

$\underline{S}$  is then transformed into a cost matrix  $\vec{C}(f, t, \bar{n}, \dot{s}, \dot{p}, \dot{f}) = C(f, t, \omega, \underline{S})$  using the summation function described in Section 3.4.2.3.1.

**3.4.2.3.5 Gaussian** Let

$$G(\varpi, N)_{tw} = e^{-\frac{1}{2} \frac{\varpi n}{W}^2}$$

---

**Algorithm 2:** Construct *contig-evolution* dissimilarity matrix by modifying  $S_{ij}$  in-place.

---

```

1  $\underline{S}_{ij} \leftarrow \hat{S}_{ij}(\hat{\Omega})$ 
2 for  $d \leftarrow 1$  to  $W$  do
3    $g \leftarrow \text{diag}(\underline{S}_{ij}, d)$ ; ( $d$  is the diagonal of  $\underline{S}_{ij}$ )
4    $s_i = \text{sign}(g)$ ; take a recording of the signs
5    $g \leftarrow \text{diff}(d, \dot{e})$ ; ( $\dot{e}$ th order differences, zero-padded at the start by  $\dot{e}$  so the size
   of  $g$  remains unchanged)
6    $g_i = g_i d^{\dot{e}}$ , for all  $i \in g$ ; apply normalization
7    $g_i = s_i g_i$ , for all  $i \in g$ ; place the original signs back
8    $\text{diag}(\underline{S}_{ij}, d) = g_i$ ; place diagonal back into  $\underline{S}_{ij}$  at  $d$ 
9  $\underline{S}_{ij} = \hat{N}(\underline{S}_{ij})$ ; normalize  $\underline{S}_{ij} = \bar{e} \underline{S}_{ij}$ ; scale by its contribution  $\bar{e}$ 
10 return  $\underline{S}$ 

```

---

for all  $n = 1, 2, \dots, W$  denote the Gaussian matrix cost function of  $N \times W$ .  $G(\varpi, N)$  is time-independent and every row is the same. This cost function will be used for regularizing the others. It could also be used on its own for comparison against a ‘naïve’ cost matrix. Increasing values of  $\varpi$  will tighten up the Gaussian. Note that values of  $\varpi \in \{1, 2, \dots, 5\}$  were used

**3.4.2.3.6 Mixing Cost Functions** Cost matrices were mixed together by adding them. In the experiments there will be a parameter for each cost matrix  $\in [0, 1]$  to show its contribution to the mixture. The cost matrices will be multiplied by this number before being mixed.

**3.4.2.3.7 Solution Shift** The estimated solution will be allowed shift in time by parameter  $\Xi$  seconds  $\in \{-5, -4, \dots, 5\}$ .

## 3.5 Computing Best Segmentation

The cost of a full segmentation is obtained by summing the costs of its tracks. The goal is now to efficiently compute the segmentation of least cost.

To reconstruct  $m$  track boundaries ( $m + 1$  tracks);

A sequence  $\mathbf{t} = (t_1, \dots, t_{m+1})$  is called an  $m/T$ -segmentation if and only if

$$1 = t_1 < \dots < t_m < t_{m+1} = T + 1.$$

$m$  is the number of tracks we are trying to find and is a parameter of the algorithm. Let us use the interpretation that track  $i \in \{1, \dots, m\}$  comprises times  $\{t_i, \dots, t_{i+1} - 1\}$ . Let  $\mathbb{S}_m^T$  be the set of all  $m/T$ -segmentations. Note that there are a very large number of possible segmentations

$$\begin{aligned} |\mathbb{S}_m^T| &= \binom{T-1}{m-1} = \frac{(T-1)!}{(m-1)!(T-m)!} = \\ &= \frac{(T-1)(T-2)\cdots(T-m+1)}{(m-1)!} \geq \left(\frac{T}{m}\right)^{m-1}. \end{aligned}$$

For large values of  $T$ , considering all possible segmentations using brute force is infeasible. For example, a two hour long show with 25 tracks would have more than

$$\left(\frac{60^2 \times 2}{25}\right)^{24} \approx 1.06 \cdot 10^{59}$$

possible segmentations.

This number can be reduced slightly by imposing upper and lower bounds on the song length. Recall that  $W$  is the upper bound (in seconds) of the song length,  $w$  the lower bound (in seconds) and  $m$  the number of tracks. With the track length restriction in place, the number of possible segmentations is still massive. A number now on the order of  $10^{56}$  for a two hour show with 25 tracks,  $w = 190$  and  $W = 60 \cdot 15$ .

Let  $N(T, W, w, m)$  be the number of segmentations with time  $T$  (in tiles),

The recursive relation

$$N(T, W, w, m) = \sum N(t_m - 1, W, w, m - 1),$$

where the sum is taken over  $t_m$  such that

$$\begin{aligned} t_m &\leq T - w + 1 & t_m &\geq T - W + 1 \\ t_m &\geq (m - 1)w + 1 & t_m &\leq (m - 1)W + 1 \end{aligned}$$

The first two inequalities mean that the length of the last track is within an acceptable boundary between  $w$  and  $W$ . The last two inequalities mean that the lengths of the first  $m - 1$  tracks are within the same boundaries.

The value of  $N(7000, 60 \cdot 15, 190, 25)$  is computed and  $5.20 \cdot 10^{56}$  is the answer; which is still infeasible to compute with brute force.

The solution presented to this problem is to find a dynamic programming recursion.

The loss of an  $m/T$ -segmentation  $\mathbf{t}$  is

$$\ell(\mathbf{t}) = \sum_{i=1}^m C(t_i, t_{i+1} - 1)$$

We want to compute

$$\mathcal{V}_m^T = \min_{\mathbf{t} \in \mathbb{S}_m^T} \ell(\mathbf{t})$$

To this end, let the recurrence

$$\mathcal{V}_1^t = C(1, t)$$

and for  $i \geq 2$

$$\begin{aligned} \mathcal{V}_i^t &= \min_{\mathbf{t} \in \mathbb{S}_i^t} \ell(\mathbf{t}) \\ &= \min_{t_i} \min_{\mathbf{t} \in \mathbb{S}_{i-1}^{t_i-1}} \ell(\mathbf{t}) + C(t_i, t) \\ &= \min_{t_i} C(t_i, t) + \min_{\mathbf{t} \in \mathbb{S}_{i-1}^{t_i-1}} \ell(\mathbf{t}) \\ &= \min_{t_i} C(t_i, t) + \mathcal{V}_{i-1}^{t_i-1} \end{aligned}$$

In this formula  $t_i$  ranges from  $t - W$  to  $t - w$ . There are  $T \times m$  values of  $\mathcal{V}_m^T$  and calculating each takes at most  $O(W)$  steps. The total time complexity is  $O(TWm)$ .

## 3.6 Confidence Intervals

It may be useful for some applications to build a framework to allow confidence intervals for our predicted indices. This may also be useful for meaningful comparison of cost matrices.

### 3.6.1 Posterior Marginal of Song Boundary

Fix a learning rate  $\eta$ , and fix  $T$  and  $m$ . Let

$$P(j, s) = \frac{\sum_{\mathbf{t} \in \mathbb{S}_m^T: t_j = s} e^{-\eta \ell(\mathbf{t})}}{\sum_{\mathbf{t} \in \mathbb{S}_m^T} e^{-\eta \ell(\mathbf{t})}}$$

That is,  $P(j, s)$  is the ‘‘posterior probability’’ that song  $j$  starts at time  $s$ .

To compute  $P(j, s)$ , an extended notion of segmentation is required. Let  $\mathbf{t}$  be a  $m/F : T$  segmentation if

$$F = t_1 < \dots < t_m < t_{m+1} = T + 1.$$

Let  $\mathbb{S}_m^{F:T}$  be the set of all  $m/F - T$ -segmentations. Let

$$\begin{aligned} \sum_{\mathbf{t} \in \mathbb{S}_m^T: t_j = s} e^{-\eta \ell(\mathbf{t})} &= \sum_{\substack{\mathbf{t} \in \mathbb{S}_{j-1}^{s-1}, \\ \mathbf{t}' \in \mathbb{S}_{m-j+1}^{s:T}}} e^{-\eta(\ell(\mathbf{t}) + \ell(\mathbf{t}'))} = \\ &= \left( \sum_{\mathbf{t} \in \mathbb{S}_{j-1}^{s-1}} e^{-\eta \ell(\mathbf{t})} \right) \left( \sum_{\mathbf{t}' \in \mathbb{S}_{m-j+1}^{s:T}} e^{-\eta \ell(\mathbf{t}')} \right) \end{aligned}$$

which upon abbreviating

$$\mathcal{H}_m^t = \sum_{\mathbf{t} \in \mathbb{S}_m^t} e^{-\eta \ell(\mathbf{t})} \quad \mathcal{T}_m^f = \sum_{\mathbf{t} \in \mathbb{S}_m^{f:T}} e^{-\eta \ell(\mathbf{t})}$$

means that

$$P(j, s) = \frac{\mathcal{H}_{j-1}^{s-1} \cdot \mathcal{T}_{m-j+1}^s}{\mathcal{H}_m^T}.$$

So it suffices to compute  $\mathcal{H}_m^t$  and  $\mathcal{T}_m^f$  for all relevant  $t$  and  $m$ . Let

$$\mathcal{H}_1^t = e^{-\eta C(1,t)} \quad \mathcal{T}_1^f = e^{-\eta C(f,T-f+1)}$$

and for  $m \geq 2$

$$\begin{aligned} \mathcal{H}_m^t &= \sum_{t_m} \sum_{\mathbf{t} \in \mathbb{S}_{m-1}^{t_m-1}} e^{-\eta(\ell(\mathbf{t}) + C(t_m, t-t_m+1))} \\ &= \sum_{t_m} e^{-\eta C(t_m, t-t_m+1)} \sum_{\mathbf{t} \in \mathbb{S}_{m-1}^{t_m-1}} e^{-\eta \ell(\mathbf{t})} \\ &= \sum_{t_m} e^{-\eta C(t_m, t-t_m+1)} \mathcal{H}_{m-1}^{t_m-1} \\ \mathcal{T}_m^f &= \sum_{t_2} \sum_{\mathbf{t} \in \mathbb{S}_{m-1}^{t_2:T}} e^{-\eta(C(f, t_2-f) + \ell(\mathbf{t}))} \\ &= \sum_{t_2} e^{-\eta C(f, t_2-f)} \sum_{\mathbf{t} \in \mathbb{S}_{m-1}^{t_2:T}} e^{-\eta \ell(\mathbf{t})} \\ &= \sum_{t_2} e^{-\eta C(f, t_2-f)} \mathcal{T}_{m-1}^{t_2} \end{aligned}$$

See Figure 3.14 for an example of the posterior for a radio show.

### 3.6.2 Posterior Marginal of Song Position

Fix a learning rate  $\eta$ , and fix  $T$  and  $m$ . Let

$$P(j, s, f) = \frac{\sum_{\mathbf{t} \in \mathbb{S}_m^T: t_j=s \wedge t_{j+1}-1=f} e^{-\eta \ell(\mathbf{t})}}{\sum_{\mathbf{t} \in \mathbb{S}_m^T} e^{-\eta \ell(\mathbf{t})}}$$

That is,  $P(j, s, f)$  is the “posterior probability” that song  $j$  starts at time  $s$  and finishes at time  $f$ . In the same vein as the last section, let

$$P(j, s, f) = \frac{\mathcal{H}_{j-1}^{s-1} \cdot e^{-\eta C(s, f-s+1)} \cdot \mathcal{T}_{m-j}^{f+1}}{\mathcal{H}_m^T}.$$

### 3.6.3 Confidence Measures

The posterior marginal of song boundaries gives estimates of confidence on reconstructed indices where there are ambiguous options. The key scenarios where the algorithm is likely to make an error of judgement due to uncertainty are; getting the time wrong or the order of tracks wrong (for example, predicting the correct time in the wrong track index).

Note that the posterior marginal of song boundaries  $P(j, s)$  contains values all in the interval  $[0, 1]$ . The sum of all times for a fixed track index (every row) in  $P(j, s)$  is 1.

#### 3.6.3.1 Index (Order)

To estimate the uncertainty of correct track alignment, the probability of all boundary placements are selected for all track indices at the predicted best time.

Let

$$\zeta_i = P(j, \Pi(\mathcal{V}_m^T, j))$$

for all  $j = 1, 2, \dots, m$  where  $\Pi(\mathcal{V}, i)$  will return the best time placement for index  $i$  in

the optimal segmentation  $\mathcal{V}$ .

Let the track index confidence measure

$$\Psi(\bar{m}) = 1 - \frac{\left(\tilde{\zeta}_{\bar{m}}\right)_2}{\left(\tilde{\zeta}_{\bar{m}}\right)_1}$$

where  $\tilde{\zeta}_i$  corresponds to  $\zeta_i$  placed into in descending order of value. Note that  $\Psi \in [0, 1]$ . The index confidence measure is a function of the ratio between the two largest values in  $P(j, s)$  for all  $j = 1, 2, \dots, m$  and at the optimal time  $s$  as estimated by the main algorithm.

### 3.6.3.2 Time

The track time confidence is estimated by the ratio of the two highest peaks in  $P(j, s)$  for all  $s = 1, 2, \dots, T$ , for all  $j = 1, 2, \dots, m$ .

Let

$$\xi(\bar{m}) = 1 - \frac{\tilde{\gamma}(\bar{m})_2}{\tilde{\gamma}(\bar{m})_1}$$

where

$$\gamma(j) = P(j, t)$$

for all  $t \in 1, 2, \dots, T$  and  $\tilde{\gamma}(j)$  are the peaks found in  $\gamma(j)$  sorted in descending order.

Note that  $\xi \in [0, 1]$

## 3.7 Experiments

### 3.7.1 Training Set

6 shows at random are selected (two of each show type) to create a training set, which will be referred to as the *GitHub training set*. See Table 3.2 to see the shows that were selected.

Table 3.2: The shows randomly selected for inclusion in the *GitHub training set*.

#	SHOW NAME	ARTIST	DATE BROADCAST
1	A State Of Trance 453	Armin Van Buuren	April 2010
2	A State Of Trance 462	Armin Van Buuren	June 2010
3	Magic Island 098	Roger Shah	March 2010
4	Magic Island 112	Roger Shah	July 2010
5	Trance Around The World 364	Above & Beyond	March 2011
6	Trance Around The World 372	Above & Beyond	May 2011

### 3.7.2 Number Of Tracks Known A Priori

The primary goal of this research is to reconstruct optimal track boundaries when the number of tracks is known a priori. This experiment will pass the actual number of tracks as a parameter into the algorithm input variable  $m$ . The advantage of this is that the number of predicted tracks will equal the number of actual tracks so intuitive measures of predictive performance can be employed.

#### 3.7.2.1 Evaluation

The inherent challenge with quantifying the performance of our approach is that if any tracks are misplaced, it may have a cascade effect. For example if one track is placed too many early on in a show, many of the subsequent tracks may be correctly detected but placed out of alignment.

For the task of computing the best cost segmentation when the number of tracks are known a priori, simple statistical descriptions of the track residuals can be used;  $|P_{st} - A_{s't'}|$  where  $P$  is the predicted track,  $A$  the actual track, for show  $s$  and track  $t$  (for all  $s, t$  and  $s', t'$  in the corpus). The mean average will give a good indication of the amount of misplacements (how robust the method is). The median of the residuals will indicate the actual track accuracy invariant to any catastrophic misplacements. The standard deviation of the residuals will indicate the spread of error.

### 3.7.2.2 Finding The Best Parameters

The GitHub (see Section 3.7.1) training set was used to find robust algorithm parameters using a stochastic optimization (genetic algorithm) search. The genetic algorithm was selected because it allows integer constraints. A population size of 50 was selected, an elite count of 7 and crossover fraction of 0.5. The stopping limit for the algorithm was when the optimization objective function had stalled for 5 generations. The `ga`<sup>3</sup> function from the MATLAB global optimization toolbox was used.

Two objective functions were considered; minimizing the mean absolute track error and secondly the median absolute track error (see Section 3.7.2.1).

For each of these objective functions 5 conditional experiments were devised involving a selection of cost matrices described in Section 3.4.2.3; sum and Gaussian, symmetry and Gaussian, contiguity and Gaussian, evolution and Gaussian, and all cost matrices allowed. When cost matrices were not allowed to participate in an experiment, their contribution variable was fixed at 0. Therefore; there are 10 experiments defined. Robust parameters were found for each experiment using the genetic algorithm as described and the parameters found are listed in Table 3.5. An experiment number has been assigned to each configuration.

### 3.7.2.3 Results

Please see Tables 3.3 and 3.4 for the results and Figure 3.15 for a histogram comparing track errors for experiments 1 to 5.

The *contig-static*, *evolution* and *symmetry* cost matrices fail to perform well on their own. It was not really the intention to design these cost matrices to work well on their own but rather to augment the summation matrix. The Gaussian cost matrix then adds regularization. As reported in [109], the sum cost matrix performs robustly independently. When the best combination of all cost matrices is considered as defined by the results of the genetic algorithm (experiments 1, 6); the overall mean performance

<sup>3</sup><http://www.mathworks.co.uk/help/gads/ga.html>

Table 3.3: These are the main results for all cost matrices with parameters optimized for the best mean absolute accuracy. The tuple  $\langle a, b, c \rangle$  is used to indicate the results where  $a$  is the median absolute error in seconds,  $b$  the mean absolute error in seconds and  $c$  the standard deviation in seconds (see Section 3.7.2.3). The experiment number is shown on the left.

		lindmik	magic	tatw	asot	all
2	CONTIG	$\langle 14, 70.3, 133.4 \rangle$	$\langle 9, 18.6, 52.6 \rangle$	$\langle 11, 34.6, 93.7 \rangle$	$\langle 13, 58.5, 121.1 \rangle$	$\langle 12, 42.9, 102.8 \rangle$
3	EVOLUTION	$\langle 14, 54.8, 110.2 \rangle$	$\langle 12, 27.1, 58.4 \rangle$	$\langle 9, 23.5, 54.6 \rangle$	$\langle 15, 50.2, 101.8 \rangle$	$\langle 12, 37.5, 83.3 \rangle$
4	SUM	$\langle 8, 34.3, 81.1 \rangle$	$\langle 8, 14, 34.3 \rangle$	$\langle 6, 16.6, 50.26 \rangle$	$\langle 8, 33.1, 77.3 \rangle$	$\langle 7, 23.7, 62.1 \rangle$
5	SYMMETRY	$\langle 10, 20.4, 49.8 \rangle$	$\langle 9, 16.6, 43.7 \rangle$	$\langle 8, 11.7, 16.8 \rangle$	$\langle 11, 26.2, 60.4 \rangle$	$\langle 9, 19, 46.3 \rangle$
1	ALL	$\langle 8, 19.5, 54.3 \rangle$	$\langle 8, 16.9, 50.9 \rangle$	$\langle 6, 8.9, 14.2 \rangle$	$\langle 7, 24.3, 58.3 \rangle$	$\langle 7, 17.6, 47.9 \rangle$

Table 3.4: These are the main results for all cost matrices with parameters optimized for the best median absolute accuracy.

		lindmik	magic	tatw	asot	all shows
7	CONTIG	$\langle 15, 96.3, 173.7 \rangle$	$\langle 9, 28.4, 78.8 \rangle$	$\langle 8, 51.5, 124.6 \rangle$	$\langle 16, 78.1, 146.7 \rangle$	$\langle 10, 60, 131.1 \rangle$
8	EVOLUTION	$\langle 20, 96.6, 149.8 \rangle$	$\langle 13, 33.9, 69.8 \rangle$	$\langle 8, 30.9, 76 \rangle$	$\langle 18, 16.6, 114.8 \rangle$	$\langle 12, 50.3, 104.5 \rangle$
9	SUM	$\langle 8, 39.9, 95 \rangle$	$\langle 8, 15.8, 42.5 \rangle$	$\langle 6, 18.6, 59.7 \rangle$	$\langle 7, 38.2, 90.2 \rangle$	$\langle 7, 27.2, 73.0 \rangle$
10	SYMMETRY	$\langle 12, 84.6, 161.8 \rangle$	$\langle 9, 19.6, 51.5 \rangle$	$\langle 9, 52.5, 127.9 \rangle$	$\langle 18, 92.1, 171.2 \rangle$	$\langle 11, 61, 135.5 \rangle$
6	ALL	$\langle 6, 13.1, 28.7 \rangle$	$\langle 9, 17.6, 44.7 \rangle$	$\langle 5, 8.9, 15 \rangle$	$\langle 6, 25.7, 62.4 \rangle$	$\langle 6, 17.4, 44.8 \rangle$

is improved substantially which means there are fewer catastrophic misplacements.

TATW has the best overall performance (median 5 seconds on experiment 6) and this is likely to indicate that it has an overall lower ‘complexity’ of DJ-mixing.

It can be concluded from these results that taking the best mixture of cost matrices significantly improves the robustness (mean-absolute of track errors) and spread (standard deviation of track errors) of the results.

On [109] disincentive-only summation matrix were effectively being used, and we reported that normalizing costs on the square root of track length produced the best result. Something similar is happening here as the genetic algorithm has selected values less than 1 on the sum normalization (experiments 4,9) which would encourage placement of longer tracks. As opposed to [109], no shows are being discarded from evaluation. To save time on the experiments the smallest tile size was set to 3, and this was the value returned for both mixtures (mean-optimized and median-optimized). Therefore it is possible that the results would improve further if the experiment went down to a tile size of 1 second.

### 3.7.2.4 Confidence Interval Analysis

See Figure 3.18 for illustrations of the time index confidence  $\xi(\bar{m})$ , index placement confidence  $\Psi(\bar{m})$  and track error residuals averaged and shown as a function of progression through the shows. See Section 3.6 for definitions. Because the number of tracks varies greatly, the confidence measures for all shows have been re-sampled into the set of indices  $1, 2, \dots, 15$  for the number of tracks. Thus it is possible to get an indication of aggregate confidence measures in relation to the progress of the show.

It can be concluded from these illustrations that the likelihood of placing the correct index statistically declines towards the middle of the shows. Perhaps this means that the summation matrix would have performed increasingly poorly as show lengths got longer. But mixing the cost matrices together in the optimal way (as dictated by the results of the evolutionary algorithm) apparently removes this tendency to a large extent.

A low index confidence would increase the probability of a *catastrophic misplacement* causing a significant deterioration on the overall mean performance metric. So these illustrations give us some insight on why mixing the cost matrices together significantly affected the overall mean evaluation metric as described in Section 3.7.2.3.

Mixing the cost matrices had little effect on the time placement of indices.

### 3.7.3 Number Of Tracks Not Known A Priori

The main goal of our research is providing the best possible time dependent contiguous segmentation given a fixed number of tracks  $m$ , rather than estimating  $m$ . This problem has not been addressed before to our knowledge; namely that the number of segments is known a priori but segmentation itself is not. However, the number of tracks could be estimated in a naïve sense because the variable of track lengths is Gaussian (see Figure 3.2).

Let us propose the following method of adapting our framework to estimate the number of contiguous segments in a data stream. For every possible candidate number

of tracks  $\Delta$ , compute the cost of fitting  $\Delta$  tracks using the algorithm described in Section 3.5 and normalize it by  $\Delta$  and take the solution  $n$  on the saddle point where the normalized quantity achieves the minimum (see Figure 3.19 for an illustration). The same genetic algorithm as described in Section 3.7.2.2 was executed to find the best set of parameters for this task. This is referred to as experiment 11 in Table 3.5.

For comparison; Foote’s method of segmentation [49] was implemented. Foote correlated a Gaussian tapered checkerboard kernel of a fixed width  $\beta$  down the diagonal of  $S$  to produce a novelty function. The multiplicative Gaussian kernel has a width and standard deviation of  $\beta$ . Any peaks found to be above a threshold  $\iota$  are counted as novelty peaks. The kernel can be constructed with the Kronecker tensor product.

To improve upon Foote’s method; a radius parameter  $\hat{R}$  introduced which adds the constraint that no two peaks are allowed to be within a radius  $\hat{R}$  of each other. This seemed like an obvious piece of domain knowledge which was to the algorithm in the interests of fairness. The peaks are found in the (time) order of the dataset. When an adjacent set of peaks are marked within  $R$ , only the first one survives. A genetic search as described in Section 3.7.2.2 was executed to find parameters that perform robustly for this task. The parameters found were  $\beta = 120, \iota = 0.3, \hat{R} = 50$ . Let us henceforth refer to this algorithm as the *enhanced* Foote novelty peak approach.

A naïve comparator of guessing how many tracks were in a show was also used, it divided the show length by the overall average track size.

See Figure 3.20 for an comparison of these three methods of track estimation. Our method estimates the correct number of tracks 45.7% of the time, the novelty peak finding approach 44.5% of the time and the naïve approach in 11.5% of cases. There is not a significant difference in performance between the enhanced version of Foote’s approach and ours. An interesting feature of Foote’s enhanced algorithm is that it almost never overestimates the number of tracks. It seems likely that some combination of the methods would yield improved results.

### 3.7.3.1 Comparison of Methods For Segmentation

It is useful to compare our algorithm for reconstructing track boundaries with Foote’s novelty peak finding method in a general sense. The drawback of Foote’s method is that it is problematic to find a fixed number of novelty peaks. It is clearly adaptable to find a *maximum* number of peaks but this does not help because it already has the interesting feature that it apparently rarely overestimates the number of tracks, it almost always underestimates.

Let us henceforth transform the problem into one of binary classification subject to a variable threshold. When a predicted boundary is within a threshold time horizon  $\tilde{t}$  of an actual boundary, it will be called a true positive. Otherwise, a true negative. Machine learning evaluators can be borrowed for binary classification problems; precision, recall and  $F_1$  score.

Let

$$F_1 = 2 \frac{P \cdot R}{P + R}$$

be the harmonic mean of precision

$$P = \frac{|TP|}{|P|}$$

and recall (true positive rate)

$$R = \frac{|TP|}{|A|}$$

subject to threshold time horizon  $\tilde{t}$ . Note  $|P|$  denotes the number of predicted tracks and  $|A|$  the number of actual tracks in a given show.

### 3.7.3.2 Results

See Figure 3.16 for a break down of  $F_1$  scores for each data set and Figure 3.17 for the overall  $F_1$  scores for the entire corpus, for all time thresholds. Note that guessing refers

to placing tracks every  $\dot{q}$  seconds until no more can be placed, where  $\dot{q}$  is the average track length. What is clear is that adding the radius enhancement to Foote’s method significantly improves its performance and it slightly out-performs our algorithm when the number of tracks apart from on the `lindmik` dataset were estimated.

Plotz [104] achieved a true positive rate of 81% within 10 second boundaries from ground truth for a similar task. Considering the standard deviation of human disagreement on annotations for dance music can be over 9 seconds (see Section 3.3), this seems unobtainable with the corpora used here. Our overall true positive average is 63% at the 10 seconds threshold. For `tatw` achieve 72.2% is achieved here which may be more like the corpora Plotz worked with.

When the number of tracks is known a priori, Foote’s method is significantly out-performed. And there is no obvious way to modify Foote’s method to select the correct number of tracks even if it is known a priori.

It should be noted that our method of track estimation may be not be optimal. The evolutionary algorithm was only trained on 6 shows on the *GitHub training set* and estimated them all correctly. This means that the parameters found could possibly have been better. Increasing the size of the training set would address this issue.

DOMAIN	NOTATION	MEAN ALL	MEAN CONTIG	MEAN EVOLUTION	MEAN SUM	MEAN SYMMETRY	MEDIAN ALL	MEDIAN CONTIG	MEDIAN EVOLUTION	MEDIAN SUM	MEDIAN SYMMETRY	TRACK ESTIMATION	
EXPERIMENT NUMBER		1	2	3	4	5	6	7	8	9	10	11	
Seconds Per Tile (S)	3, ..., 50	$M$	3	3	6	5	8	3	7	3	5	9	38
Min. Track Length (S)	80, ..., 180	$w$	167	146	108	165	140	88	98	173	94	85	155
Max. Track Length (S)	600, ..., 900	$W$	691	879	897	894	811	631	801	889	642	635	619
Bandwidth (Hz)	1, ..., 15	$b$	1	2	2	2	3	2	4	4	2	5	3
Low Pass Filter (Hz)	800, ..., 1950	$l$	1039	1912	1893	1387	874	888	1065	1206	1880	1005	1019
High Pass Filter (Hz)	50, ..., 500	$h$	62	73	81	69	54	55	54	70	75	51	201
Solution Shift (S)	-3, ..., 5	$\Xi$	-1	5	-3	-2	2	-4	-2	-2	-2	-1	5
Cosine Normalization	[0.4, 1.4]	$\hat{c}$	1.17	0.77	0.92	1.19	1.36	0.88	0.71	0.73	1.15	1.14	0.98
Sum Contribution	[0, 1]		0.99			0.81		0.77		0.63			0.55
Sum Normalization	[0, 1]	$\Omega$	1.36			0.73		1.11		0.47			0.71
Sum Incentive	[0, 1]	$\omega$	0.68			0.52		0.23		0.30			0.05
Gaussian Contribution	[0, 1]		0.52	0.08	0.69	0.17	0.29	0.63	0.11	0.51	0.08	0.02	0.15
Gaussian Incentive	[0, 1]		0.85	0.82	0.43	0.47	0.56	0.10	0.14	0.40	0.85	0.54	0.53
Gaussian Width	1, ..., 4	$\varpi$	1	1	1	4	1	1	2	1	1	2	4
Evolution Contribution	[0, 1]	$\bar{e}$	0.05		0.63			0.49		0.35			0.48
Evolution Incentive	[0, 1]	$\bar{\Omega}$	0.53		0.60			0.15		0.66			0.71
Evolution Normalization	[0.1, 3]	$\hat{n}$	1.30		0.08			1.10		0.06			1.79
Evolution Diff. Order	1, ..., 50	$\hat{e}$	45		1			7		40			16
Contig Past Contribution	[0, 1]	$\bar{p}$	0.50	0.10				0.62	0.69				0.27
Contig Past Diff. Order	1, ..., 50	$\bar{p}$	13	44				41	46				30
Contig Past Incentive	[0, 1]	$\bar{\Omega}$	0.50	0.24				0.95	0.53				0.98
Contig Normalization	[0.1, 3]	$\bar{n}$	0.74	0.26				1.60	0.33				1.91
Contig Future Contribution	[0, 1]	$\bar{f}$	0.59	0.81				0.54	0.27				0.95
Contig Future Diff. Order	1, ..., 50	$\bar{f}$	35	3				30	21				45
Contig Future Incentive	[0, 1]	$\bar{\Omega}$	0.08	0.24				0.60	0.89				0.96
Symmetry Contribution	[0, 1]		0.18			0.66	0.11				0.98	0.19	
Symmetry Incentive	[0, 1]	$\bar{\Omega}$	0.16			0.78	0.24				0.45	0.26	
Symmetry Normalization	[0.1, 3]	$\bar{\omega}$	0.73			0.77	0.72				0.55	1.09	

Table 3.5: Results for stochastic optimization (evolutionary algorithm) search of parameter space. Note that the search space  $T$  was limited to a minimum of 3 seconds to save computation time.

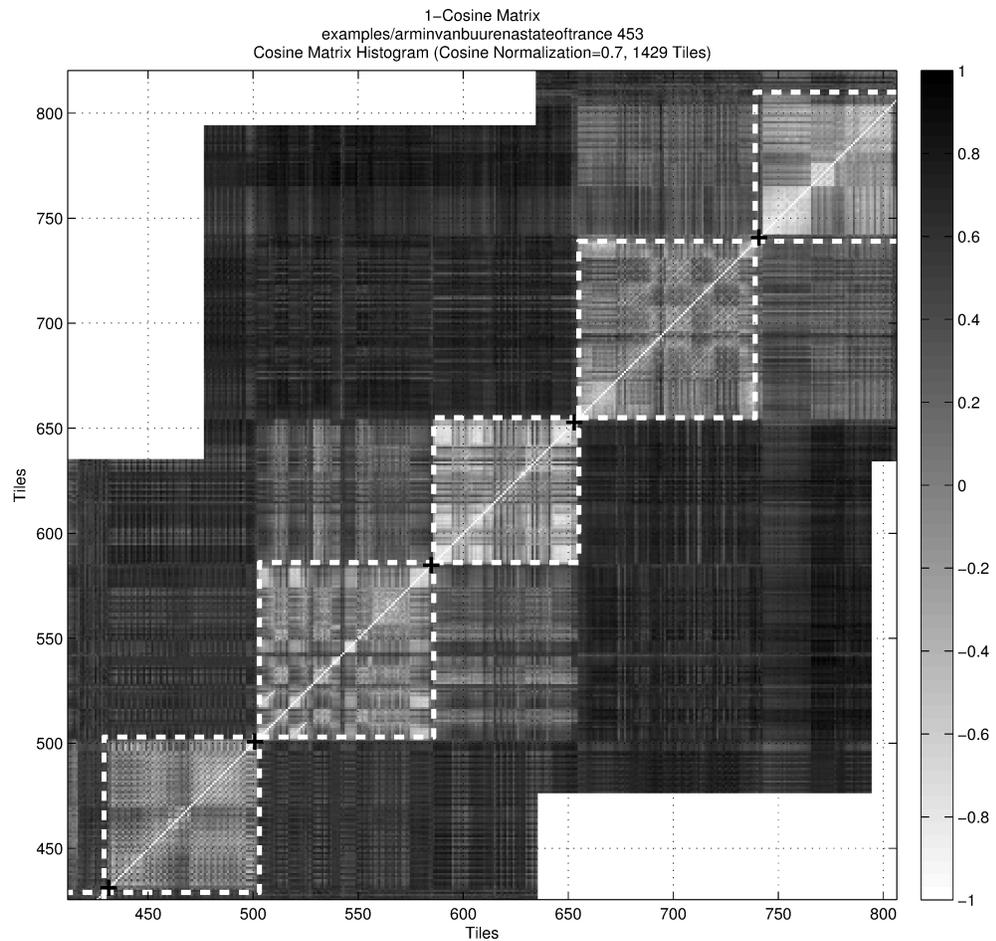


Figure 3.11: An illustration of the similarity matrix  $S$  (cosines) with the actual indices drawn on with black crosses, and our reconstructed annotations indicated with the dotted white lines. Note that to save time on the computation the entire matrix is not calculated; which is why there are some empty regions on the corners.

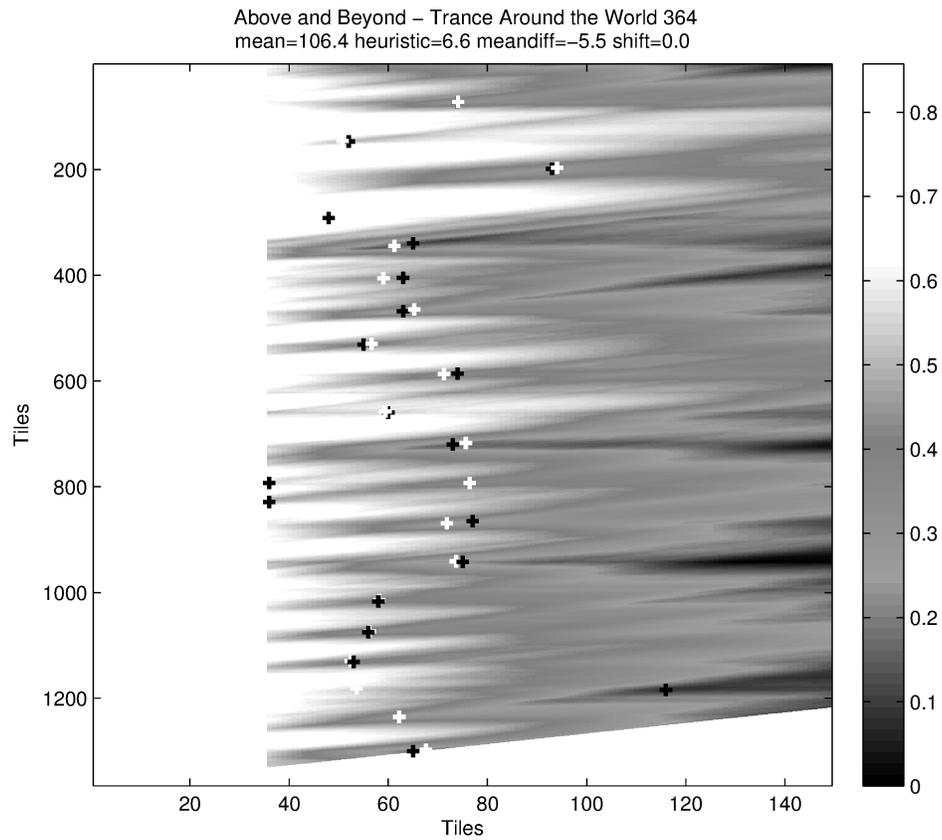


Figure 3.12: Summation cost matrices for Magic Island episode 110 with an incentive bias  $\Omega = 1$  and therefore containing disincentives.

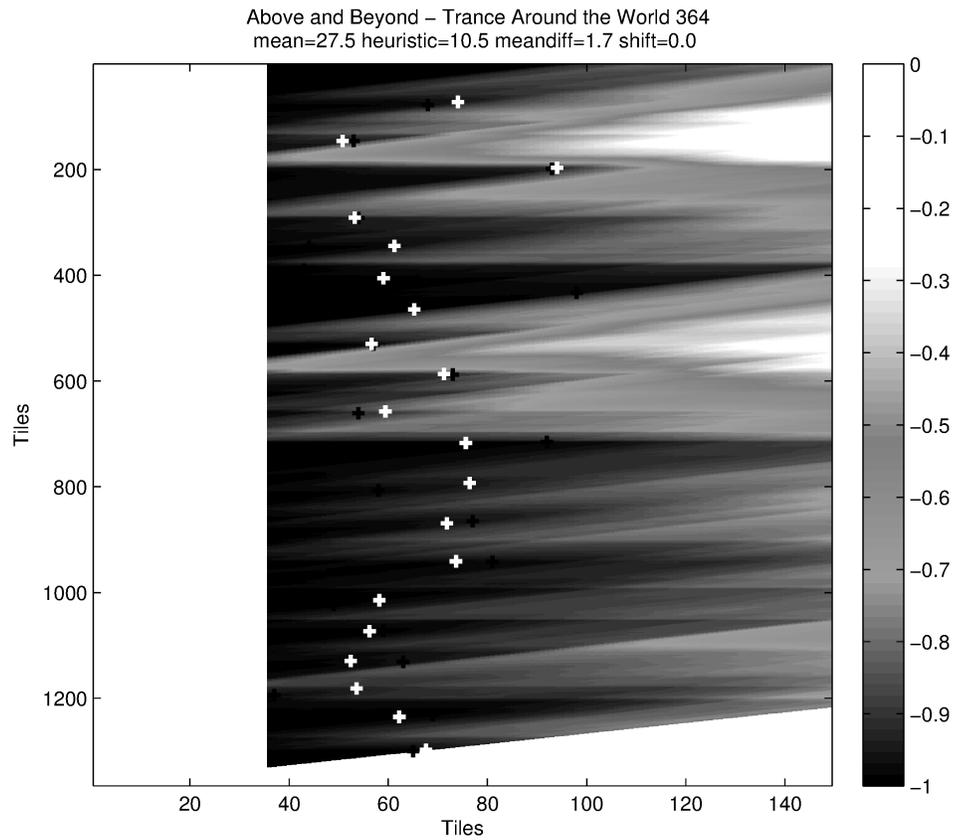


Figure 3.13: Summation cost matrices for Magic Island episode 110 with an incentive bias  $\Omega = 0$  and therefore containing incentives.

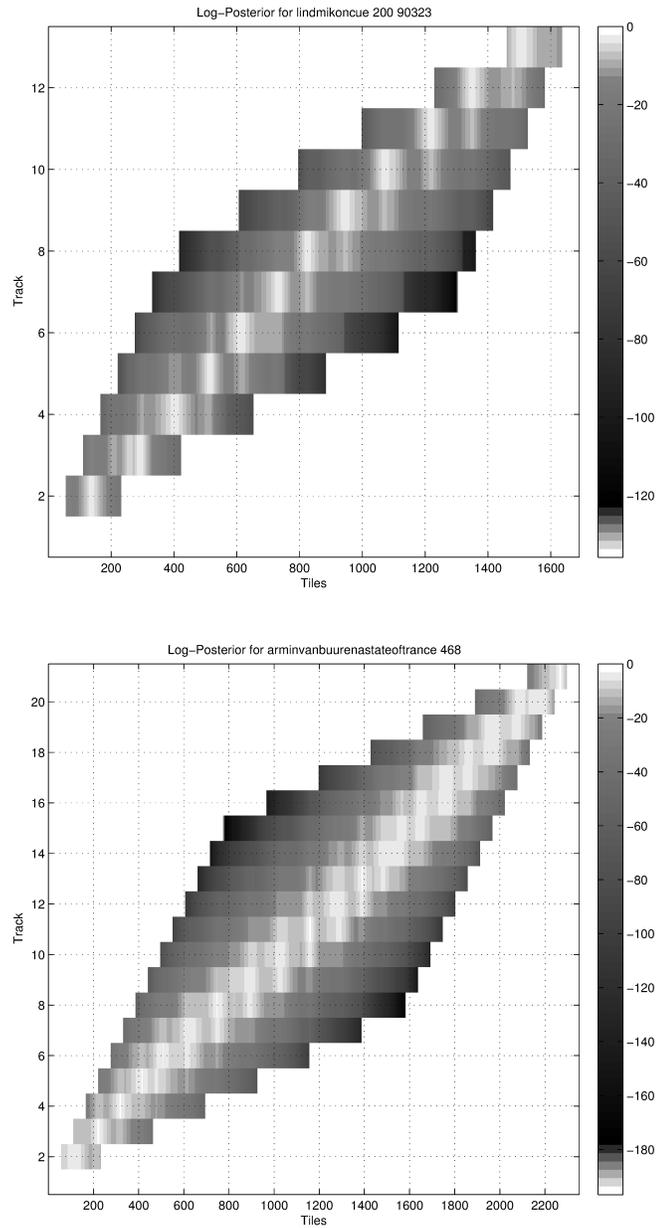


Figure 3.14: A visualization of  $\log(P(j, s))$  ( $\eta = 10$ ) for two of the shows in the training set. Ostensibly; uncertainty pertaining to the correct time and index (i.e. track number 2, 3, 4) placement increases towards the middle of the shows.

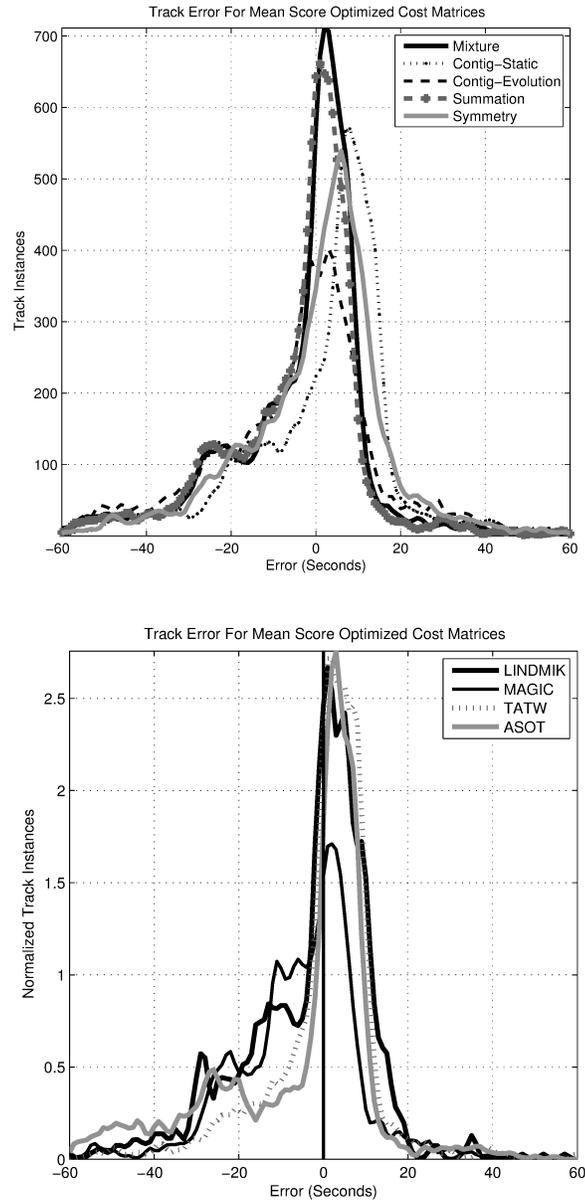


Figure 3.15: Histogram of the residuals (errors) between reconstructed and human captured time indices per experiment (top) and with best mean-optimized mixture broken down by show (bottom). Apart from obvious noise there appears to be a tendency for the algorithm to place an index slightly earlier.

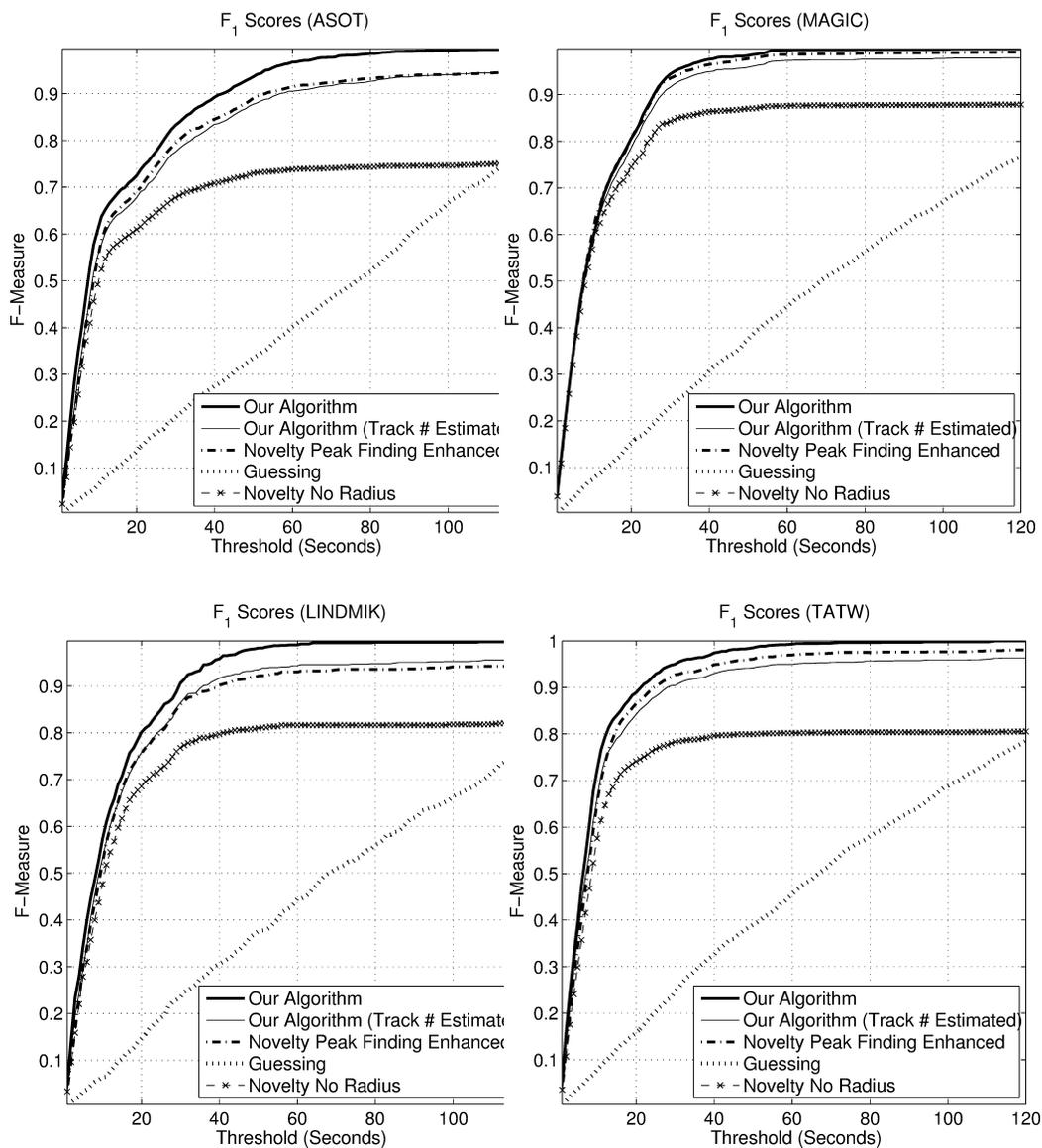


Figure 3.16: Comparison of the  $F_1$  scores against time thresholds on the 4 data sets. On the *lindmik* dataset where the number of tracks is highly unpredictable, our method combined with track estimation beats Foote’s *enhanced* novelty method at higher thresholds.

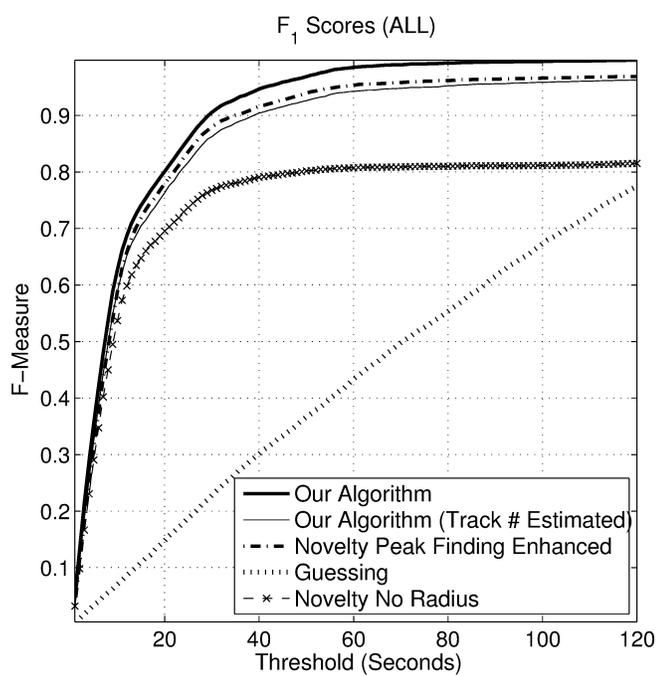


Figure 3.17: Comparison between our algorithm and the Foote novelty peak finding approach on all of the datasets.

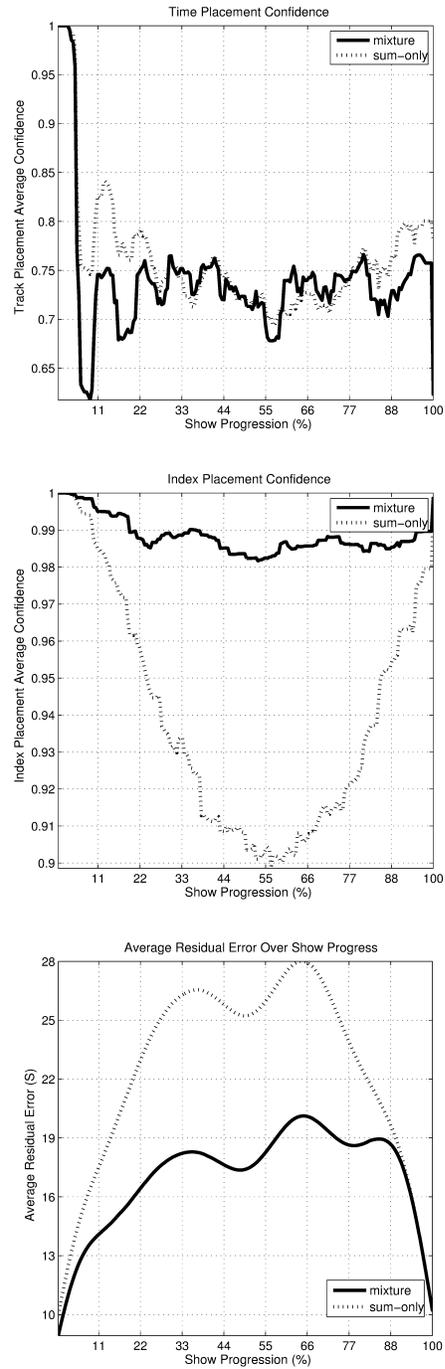


Figure 3.18: Confidence intervals and error residuals averaged over show progression.

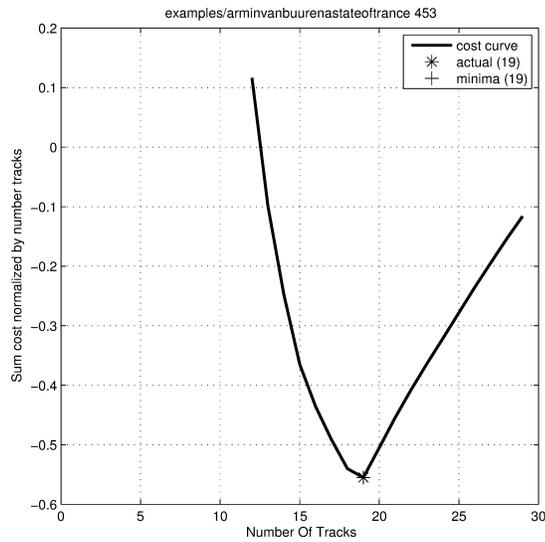


Figure 3.19: Number of tracks estimated correctly a show in the GitHub training set after a genetics algorithm was executed to select a robust set of algorithm parameters.

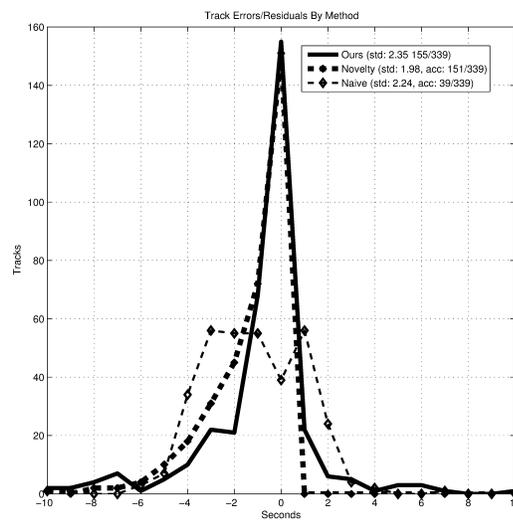


Figure 3.20: Track estimation error on our method as described in Section 3.7.3, Foote's novelty function and naïve guessing.

## Chapter 4

# Forecasting Framework

In this chapter, the on-line forecasting protocol, the aggregating algorithm, sleeping specialist experts and fixed- and variable-share algorithms are discussed in detail.

### 4.1 Online Protocol/Regression

We consider the on-line protocol where on each trial  $t = 1, 2, \dots$  the learner observes a signal  $x_t$  and attempts to predict an outcome  $y_t$ , which is shown to the learner later. The performance of the learner is measured by means of the cumulative loss; throughout all of the thesis we use square loss.

This problem has been addressed within the context of on-line regression. One option is to apply traditional batch algorithms such as ridge regression in the on-line context; see [135] for a comparison of losses of ridge regression in the on-line and batch modes. An alternative is to develop special on-line regression algorithms. For example, see [6, 128], and Section 11.8 of [23] for aggregating algorithm regression also known as the Vovk-Azoury-Warmuth predictor. An important area is the development of regression algorithms targeted at changing dependencies; see [64] for tracking algorithms, [19] for aggregating algorithm regression with changing dependencies, and [30] for regression under discounted loss.

The regression algorithms are sometimes time-consuming as they involve dealing with a large matrix; kernel algorithms normally involve a matrix of size  $t \times t$ . One can save time by developing incremental update tricks (see Subject 4.2.0.3) or using a sliding window approach with a fixed window size.

A more important drawback is the need for smoothness: regression algorithms require the dependency between  $ys$  and  $xs$  to stay the same or to change very slowly. The levels of variability acceptable for time series techniques are rarely suitable for regression methods.

## 4.2 Kernel Ridge Regression

Ridge regression is a powerful technique of machine learning. It was introduced in [67]; the kernel version of it is derived in [108] and some interesting derivations are presented in [135]. Kernel ridge regression can be used in the batch or on-line setting. In dual variables the kernel trick (also used in support vector methods) can be employed to facilitate regression in high or infinite feature space while remaining a bare bones linear classifier under the covers. Cholesky matrix update tricks allow the online task to be performed in linear time.

Suppose we are given a set of  $T$  examples  $(x_1, y_1), (x_2, y_2), \dots, (x_T, y_T)$ , where  $x_i \in \mathbb{R}^n$  are *signals* and  $y_i \in \mathbb{R}$  are *outcomes* or *labels*. We want to find a dependency between signals and outcomes and to be able to predict  $y$  given a new  $x$ . This problem is often referred to as the *regression problem*.

Let us take  $a \geq 0$  and consider the expression for the primary form of ridge regression

$$L_{\text{RR}}(w) = a\|w\|^2 + \sum_{i=1}^T (w'x_i - y_i)^2.$$

The  $w$  that minimises this sum is the solution of RR. The first term is a regularising control which has the effect of flattening or simplifying the fit of  $w$  in the spirit of “Occum’s Razor” originating from the 14th century logician and Franciscan friar

William of Ockham. The idea is that if you learn the data too tightly you will perform poorly on new, unseen data. This concept is referred to as *overfitting* (see [7, 41, 63] for more information).

The closed form solution to the primary of RR is  $w = (aI + X'X)^{-1}X'Y$ , and using the matrix identity  $A(aI + A'A)^{-1} = (aI + AA')^{-1}A$  we can rewrite the ridge regression solution as follows [108]. For an arbitrary  $x \in \mathbb{R}^n$  the outcome suggested by ridge regression is  $w'x$  and this can be rewritten as

$$\begin{aligned} w'x &= ((aI + X'X)^{-1}X'Y)'x, \\ &= Y'X(aI + X'X)^{-1}x, \\ &= Y'(aI + XX')^{-1}Xx. \end{aligned}$$

This formula is called the dual form of the ridge regression solution.

In the spirit of the kernel trick introduced by Aizerman et al. in [3] the inner product space  $XX'$  can be replaced with a non-linear kernel allowing a linear regression method to work inside a potentially infinite feature space.

$$w'x = Y'(aI + K)^{-1}k,$$

where  $K$  is the matrix of mutual scalar products

$$K = \begin{pmatrix} \langle x_1, x_1 \rangle & \langle x_1, x_2 \rangle & \dots & \langle x_1, x_T \rangle \\ \langle x_2, x_1 \rangle & \langle x_2, x_2 \rangle & \dots & \langle x_2, x_T \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle x_T, x_1 \rangle & \langle x_T, x_2 \rangle & \dots & \langle x_T, x_T \rangle \end{pmatrix}$$

and  $k = k(x)$  is the vector of scalar products of  $x_i$  by  $x$ :

$$k = \begin{pmatrix} \langle x_1, x \rangle \\ \langle x_2, x \rangle \\ \dots \\ \langle x_T, x \rangle \end{pmatrix}.$$

Of immediate relevance to practical applications is that the kernel trick allows us to approximate non-linear relationships between signals and labels for example using Vapnik's polynomial kernel of degree  $d$ , i.e.,  $\mathcal{K}(x_1, x_2) = (\langle x_1, x_2 \rangle + 1)^d$ .

The radial basis function kernel (see [99, 110, 110]) is also commonly used in the literature and is as follows. For any  $\sigma \neq 0$ , the function  $\mathcal{K}(x_1, x_2) = e^{-\|x_1 - x_2\|^2 / (2\sigma^2)}$ , where  $\|\cdot\|$  is the Euclidean norm, is a kernel on  $\mathbb{R}^n$ . This kernel maps the input space onto the surface of an infinite dimensional unit hypersphere.  $\sigma$  acts as a regularisation parameter, with larger values leading to a smoother decision surface.

#### 4.2.0.3 Cholesky Decomposition

Ridge regression involves calculating the expression  $Y'(K + aI)^{-1}k$ , where  $K$  is a symmetric positive semi-definite matrix,  $a$  is a positive constant, and thus  $K + aI$  is symmetric positive definite.

It is usually unwise to invert a matrix explicitly. If one needs to calculate the vector  $y = M^{-1}x$ , it is better to write the formula as  $My = x$  and find  $y$  by solving the system of linear equations.

If  $M$  is symmetric positive definite, there is a convenient technique called Cholesky decomposition. The matrix  $M$  can be represented as  $LL'$ , where  $L$  is a lower triangular matrix and thus solving the equation  $My = x$  reduces to solving two equations  $Lu = x$  and  $L'v = u$  both having triangular matrices.

Finding the Cholesky decomposition takes  $O(n^3)$  arithmetic operations, where  $n \times n$  is the size of  $M$ . Solving a linear equation with a lower triangular matrix takes  $O(n^2)$

operations, as only back substitution is required.

Suppose that we apply ridge regression in on-line fashion. If we use a growing window technique, then the matrix  $K + aI$  increases on every step: the matrix for step  $n+1$  has the matrix from step  $n$  at its upper left corner with one extra line and column added. If we use a sliding window technique, then the matrix size stays the same: we need to take the matrix from step  $n$ , remove its first line and column and add one line below and one column on the right.

Spending time  $O(n^3)$  on every step recalculating the Cholesky decomposition is a grim prospect, especially if the matrix size is in hundreds, so incremental update techniques are required.

#### 4.2.0.4 An Identity

Let  $L$  be a square lower triangular matrix of size  $n \times n$ . Consider a partitioning

$$L = \begin{pmatrix} Z & 0 \\ Y' & X \end{pmatrix}, \quad (4.1)$$

where  $Z$  and  $X$  are square lower triangular matrices and  $0$  is a (perhaps non-square) matrix of zeroes. Calculating the product of block matrices yields

$$LL' = \begin{pmatrix} Z & 0 \\ Y' & X \end{pmatrix} \begin{pmatrix} Z' & Y \\ 0 & X' \end{pmatrix} = \begin{pmatrix} ZZ' & ZY \\ Y'Z' & Y'Y + XX' \end{pmatrix}. \quad (4.2)$$

We will be using this identity to obtain incremental methods.

#### 4.2.1 Normalisation

It is good practice to normalise or standardise the data prior to applying an algorithm to it. Features that are too big can cause computational problems and a feature that is consistently much larger than another one may be given undue extra importance.

The signals  $x_i$  are linearly scale transformed onto the  $[-1, 1]$  interval using the function

$$x_i = \frac{\frac{(x_i - \min(x_i))}{\max(x_i) - \min(x_i)} + 1}{2}.$$

The  $[-1, 1]$  interval has been selected because it is optimal for Vapnik's polynomial kernel.

### 4.3 Growing Window

We start with the growing window case. Let us partition a symmetric positive definite  $(n + 1) \times (n + 1)$ -matrix  $M$  as

$$M = \begin{pmatrix} C & B \\ B' & A \end{pmatrix}, \quad (4.3)$$

where the size of  $C$  is  $n \times n$  and the size of  $A$  is  $1 \times 1$ . Consider  $L$  of size  $(n + 1) \times (n + 1)$  partitioned as in (4.1) with  $Z$  of size  $n \times n$ . We need to solve the matrix 'equation'

$$\begin{pmatrix} ZZ' & ZY \\ Y'Z' & Y'Y + XX' \end{pmatrix} = \begin{pmatrix} C & B \\ B' & A \end{pmatrix}. \quad (4.4)$$

The matrix  $Z$  provides a Cholesky decomposition of  $C$ ; this is what was found on step  $n$  of regression. Note that if  $M$  is symmetric positive definite, then  $C$  is always symmetric positive definite and  $Z$  is non-singular.

In order to find  $Y = Z^{-1}B$  we need to solve the equation  $ZY = B$ , which takes time  $O(n^2)$  as  $Z$  is lower triangular. For  $X$ , which actually consists of a single number

$X = (x)$ , we have the equation  $Y'Y + XX' = A$ , i.e.,

$$\begin{aligned} x^2 &= A - YY' \\ &= A - B'(Z')^{-1}Z^{-1}B \\ &= A - B'(ZZ')^{-1}B \\ &= A - B'C^{-1}B . \end{aligned}$$

This expression is called the Schur complement of  $C$  in  $M$  and it can be shown to be positive definite (see, e.g., [68], Section 7.7.6). Therefore the number  $A - YY'$  is always positive and we can calculate  $x$  by taking the square root.

Thus the update is always possible and can be performed using  $O(n^2)$  operations (counting the square root as an elementary operation).

Note that as a by-product we have shown that Cholesky decomposition of positive semi-definite matrices always exists and can be obtained using  $O(n^3)$  operations.

## 4.4 Sliding Window

The matrix for step  $n + 1$  of the sliding window can be obtained in two steps, first, adding one row below and one column on the right and, secondly, removing the top row and left-most column.

The first step is identical to the growing window. We have shown that we can update the Cholesky decomposition for the larger matrix using  $O(n^2)$  operations. Now we need to consider the second step and get the decomposition for a smaller matrix.

Let us partition a symmetric positive definite  $(n + 1) \times (n + 1)$ -matrix as in (4.3), where  $C$  is of size  $1 \times 1$  and  $A$  is of size  $n \times n$ . Suppose that we know a decomposition  $L$  of  $M$  and want to obtain a decomposition of  $A$ . Partitioning  $L$  as in (4.1) with  $X$  of size  $n \times n$ , we get (4.4) and  $A = Y'Y + XX'$ .

Our problem has reduced to the following. Knowing a lower triangular  $X$  and

a vector  $Y$  we need to obtain a Cholesky decomposition of  $Y'Y + XX'$ . This is a well-known problem known as low-rank update. Quadratic methods for solving it are discussed in, e.g., [13] and [54]. MATLAB has a function `cholupdate`.

Thus the update for the sliding window is possible and can be done in quadratic time.

## 4.5 Merging Decision Strategies

*Making rational decisions is a central problem in science and everyday life. (Polynomials of which degree should I use to fit my data sets? Should I take my umbrella today, tomorrow, etc.? Which stocks should I buy and sell this year?) Only rarely we can readily choose the best course of action; more often we will have a more or less extensive (maybe infinite) family of potentially successful decision strategies. (Whether a decision strategy is successful will depend not only on the merits of this strategy but also on the future events which we do not know yet.) However, at the end of the day we must choose one specific decision strategy, so we naturally arrive at this problem: given a family of decision strategies, find a new decision strategy which will perform, under any circumstances, almost as well as the best (under those circumstances) decision strategy in the family. Vovk, 2001 [128].*

Aspiring to be nearly as good as the best expert might not seem ambitious enough. However, given the theoretical bounds that could still apply to any practical situation; it is actually a strong statement.

### 4.5.1 Prediction Framework

This thesis is concerned with prediction in the following framework. Let outcomes  $\omega_1, \omega_2, \dots$  from an *outcome set*  $\Omega$  occur successively in discrete time. A *learner* tries to

predict each outcome and outputs a prediction  $\gamma_t$  from a *prediction set*  $\Gamma$  on the basis of a *signal*  $x_t$  from a *domain*  $X$  each time before it sees the outcome  $\omega_t$ . The quality of predictions is assessed by means of a *loss function*  $\lambda : \Gamma \times \Omega \rightarrow [0, +\infty]$ .

There is also a pool  $\Theta$  of (*static*) *experts*; in this thesis we assume that  $\Theta$  is finite. The experts try to predict the outcomes from the same sequence and their predictions  $\gamma_t(\theta)$  are made available to the learner.

The framework can be summarised in the following protocol:

---

**Algorithm 3:** Basic prediction

---

```

1 for  $t = 1, 2, \dots$  do
2   | nature announces  $x_t \in X$ 
3   | learner outputs  $\gamma_t \in \Gamma$ 
4   | nature announces  $\omega_t \in \Omega$ 
5   | learner suffers loss  $\lambda(\gamma_t, \omega_t)$ 
6 end

```

---

Over  $T$  trials the learner suffers the cumulative loss

$$\text{Loss}_T = \sum_{t=1}^T \lambda(\gamma_t, \omega_t) .$$

We will denote the loss of the learner by  $\text{Loss}_T$  or by  $\text{Loss}_T(\text{Algorithm})$  with Algorithm in brackets being the name of the algorithm the learner uses.

In this thesis we are mostly interested in the case where the prediction and outcome spaces are an interval  $\Omega = \Gamma = [A, B]$  and the square loss function  $\lambda(\gamma, \omega) = (\gamma - \omega)^2$  is used.

### 4.5.2 Prediction with Expert Advice

This section discusses prediction with expert advice; see [23] for a complete overview. The problem of prediction with expert advice can be summarised as follows.

Suppose that there is a pool  $\Theta$  of (*static*) *experts*; throughout this thesis we assume that  $\Theta$  is finite. The experts try to predict the outcomes from the same sequence and

their predictions  $\gamma_t(\theta)$  are made available to the learner before it outputs its own.

The goal of the learner is to suffer total loss comparable to the best expert in the pool in some sense. A desirable property is to not make any assumptions about the stochastic mechanism generating the underlying data.

---

**Algorithm 4:** Prediction with expert advice

---

```

1 for  $t = 1, 2, \dots$  do
2   | experts  $\theta \in \Theta$  announce predictions  $\gamma_t(\theta) \in \Gamma$ 
3   | learner outputs  $\gamma_t \in \Gamma$ 
4   | nature announces  $\omega_t \in \Omega$ 
5   | each expert  $\theta \in \Theta$  suffers loss  $\lambda(\gamma_t(\theta), \omega_t)$ 
6   | learner suffers loss  $\lambda(\gamma_t, \omega_t)$ 
7 end

```

---

Over  $T$  trials each expert  $\theta$  suffers the cumulative loss

$$\text{Loss}_T(\theta) = \sum_{t=1}^T \lambda(\gamma_t(\theta), \omega_t)$$

and the learner suffers the cumulative loss

$$\text{Loss}_T = \sum_{t=1}^T \lambda(\gamma_t, \omega_t) ;$$

one wants the inequality  $\text{Loss}_T \lesssim \text{Loss}_T(\theta)$  to hold for all  $T = 1, 2, \dots$  and  $\theta \in \Theta$ .

It is in the spirit of prediction with expert advice not to impose any restrictions on the law generating outcomes  $\omega_t$  or on the internal working of experts. As a matter of fact, ‘nature’ and ‘experts’ are just names for the slots in the protocol.

### 4.5.3 Aggregating Algorithm

In this section we overview the standard aggregating algorithm (AA) for prediction with expert advice after [126, 128]. It takes the following parameters: a learning rate  $\eta \in (0, +\infty)$  and an initial distribution over the set of static experts  $\theta$ ; a distribution can be represented by an array of initial weights  $p_0(\theta), \theta \in \Theta$ .

The algorithm maintains an array of weights  $w_t(\theta), \theta \in \Theta$ . Their initial values are  $w_0(\theta) = p_0(\theta), \theta \in \Theta$ , and they are updated according to the rule

$$w_t(\theta) = w_{t-1}(\theta)e^{-\eta\lambda(\gamma_t(\theta), \omega_t)} = p_0(\theta)e^{-\eta\text{Loss}_t(\theta)} .$$

On step  $t$  upon observing the experts' predictions  $\gamma_t(\theta)$  the learner outputs a prediction  $\gamma_t$  that for every possible  $\omega \in \Omega$  satisfies the condition

$$\lambda(\gamma_t, \omega) \leq c(\eta)g(\omega) , \quad (4.5)$$

where

$$g(\omega) = -\frac{1}{\eta} \ln \frac{1}{\sum_{\theta \in \Theta} w(\theta)} \sum_{\theta \in \Theta} w(\theta) e^{-\eta\lambda(\gamma_t(\theta), \omega)} \quad (4.6)$$

and  $c(\eta)$  is a constant specified by the loss function  $\lambda$ . The constant is defined in such a way that  $\gamma_t$  can always be found. One can show by induction (see Lemma 1 from [128]) that

$$\sum_{t=1}^T g(\omega_t) = -\frac{1}{\eta} \ln \sum_{\theta \in \Theta} p_0(\theta) e^{-\eta\text{Loss}_T(\theta)}$$

and therefore

$$\text{Loss}_T(AA) = \sum_{t=1}^T \lambda(\gamma_t, \omega_t) \leq c(\eta) \text{Loss}_T(\theta) + \frac{c(\eta)}{\eta} \ln 1/p_0(\theta) . \quad (4.7)$$

The aggregating algorithm thus performs nearly as well as the best expert losswise.

If the prediction and outcome spaces are an interval  $\Omega = \Gamma = [A, B]$  and the square loss function is  $\lambda(\gamma, \omega) = (\gamma - \omega)^2$ , then for  $\eta$  satisfying  $0 < \eta \leq \frac{2}{(B-A)^2}$  we have  $c(\eta) = 1$  (see [29, 128]) and therefore the optimal value is  $\eta = \frac{2}{(B-A)^2}$ . For these values of  $\eta$  we can use a simple *substitution function*

$$\gamma = \frac{A+B}{2} - \frac{g(B) - g(A)}{2(B-A)}$$

mapping  $g$  to a  $\gamma$  satisfying (4.5).

The aggregating algorithm generalises Bayesian mixtures of probabilistic hypothesis (see, e.g., Section 2 of [14]); it is identical to the Bayesian mixture for the so called logarithmic loss. However it is more general in that it admits arbitrary loss functions such as the square loss function.

#### 4.5.4 Sleeping and Specialist Experts

Suppose that an expert in the prediction with expert advice framework can abstain from making a prediction on step  $t$ ; if it does so, we say that it sleeps on step  $t$ . One may want to obtain an equivalent of bound (4.7) ensuring that the learner competes well with every expert  $\theta$  on the steps where  $\theta$  is awake.

The concept of a specialist expert was proposed in [50]. In this thesis we will be discussing specialist experts as a special case of the theory of expert evaluators after [26, 27]. A simple extension of the AA may be used for specialist experts.

If an expert  $\theta$  sleeps on step  $t$ , let us assume that it suffers notional loss  $\lambda(\gamma_t(\theta), \omega_t)/c(\eta)$  (if  $c(\eta) = 1$  we can simply say that it ‘goes with the crowd’ and subscribes to yet unknown  $\gamma_t$  whatever it is going to be) and apply the AA. The weight of a sleeping expert is updated according to this notional loss. If  $\Theta_{\text{sleep}}$  is the set of experts sleeping on step  $t$  and  $\Theta_{\text{awake}}$  is the set of experts that are awake (not sleeping) on step  $t$ , then (4.5) becomes

$$\lambda(\gamma_t, \omega) \leq -\frac{c(\eta)}{\eta} \ln \frac{1}{\sum_{\theta \in \Theta} w_{t-1}(\theta)} \left( \sum_{\theta \in \Theta_{\text{awake}}} w_{t-1}(\theta) e^{-\eta \lambda(\gamma_t(\theta), \omega)} + \sum_{\theta \in \Theta_{\text{sleep}}} w_{t-1}(\theta) e^{-\eta \lambda(\gamma_t, \omega)/c(\eta)} \right)$$

or

$$e^{-\eta\lambda(\gamma_t, \omega)/c(\eta)} \sum_{\theta \in \Theta} w_{t-1}(\theta) \geq \sum_{\theta \in \Theta_{\text{awake}}} w_{t-1}(\theta) e^{-\eta\lambda(\gamma_t(\theta), \omega)} + \sum_{\theta \in \Theta_{\text{sleep}}} w_{t-1}(\theta) e^{-\eta\lambda(\gamma_t, \omega)/c(\eta)} .$$

Clearly, all terms corresponding to sleeping experts cancel out and we get

$$\lambda(\gamma_t, \omega) \leq \frac{c(\eta)}{\eta} \ln \frac{1}{\sum_{\theta \in \Theta_{\text{awake}}} w_{t-1}(\theta)} \sum_{\theta \in \Theta_{\text{awake}}} w_{t-1}(\theta) e^{-\eta\lambda(\gamma_t(\theta), \omega)} ,$$

i.e., the formula is identical to that from the aggregating algorithm except that the sum is taken over the experts that are awake. Note that we still need to update the weights of sleeping experts so that they get the right weights when they wake up. We will call this algorithm aggregating algorithm with sleeping experts (AAS).

Arguing as in the case of the AA, we get a bound similar to (4.7); by dropping equal terms in the losses on the left- and right-hand side we obtain

$$\text{Loss}_T^{(\theta)}(\text{AAS}) \leq c(\eta) \text{Loss}_T^{(\theta)}(\theta) + \frac{c(\eta)}{\eta} \ln 1/p_0(\theta) , \quad (4.8)$$

where the sum in  $\text{Loss}^{(\theta)}$  is taken only over steps when expert  $\theta$  was not sleeping.

#### 4.5.5 Switching Experts

Two key algorithms for tracking the best expert, fixed-share and variable-share, were developed and analysed in [65]. However we will follow the treatment of the topic in [127] because the unified view of [127] is easier to generalise. An informative introduction into various schemes of switching experts is *combining strategies efficiently: high-quality decisions from conflicting advice* by Koolen [82].

Bound (4.7) ensures that the AA competes well with any static expert. Suppose that we want to compete with the following *dynamic experts* instead. A dynamic expert

follows one of the static experts on each trial but it can change its allegiance and switch to some other expert between trials. We assume that it sticks to the same static expert for a while and the number of switches is relatively small compared to the total number of trials.

We can apply AA to the problem of merging dynamic experts in the following natural way. Let us emulate a family of dynamic experts with a certain initial distribution and apply AA to them. The mixture will compete well with any dynamic expert [64]. We will call the aggregating algorithm applied in this fashion aggregating algorithm for dynamic experts (AAD).<sup>1</sup>

The notation is greatly simplified with the probabilistic approach of [127]. Let us look at the pool of experts  $\Theta$  with the initial distribution on experts as at a probability space. The experts' predictions  $\gamma_t(\theta)$  and losses  $\text{Loss}_t(\theta)$  can be thought of as random variables. We will speak of them as the prediction and the loss of the stochastic predictor and use the notation  $\gamma_t(SP)$  and  $\text{Loss}_t(SP)$  respectively. Instead of describing the pool of dynamic experts and the initial distribution explicitly we can now use the probabilistic language.

Bound (4.7) becomes a special case of the bound

$$\text{Loss}_t(\text{AAD}) \leq c(\beta) \mathbf{E} \beta^{\text{Loss}_t(SP)} .$$

Particular special cases can be derived from this bound using the following proposition (Corollary 2 in [127]):

**Proposition 4.5.1.** *For every  $t = 1, 2, \dots$  every exponential learning rate  $\beta \in (0, 1)$ , and every  $L \geq 0$ , if  $\text{Loss}_t(SP) \leq L$  with probability (at least)  $p > 0$ , then*

$$\text{Loss}_t(\text{AAD}) \leq c(\beta)L + \frac{c(\beta)}{\ln(1/\beta)} \ln(1/p) .$$

Clearly, the higher the probability of a group of dynamic experts, the tighter is the

---

<sup>1</sup>In Machine Learning we want efficient strategies, this algorithm is intractable.

bound.

Let us now describe the fixed- and variable-share algorithms. The fixed-share takes a parameter  $\alpha$  called the switching rate. Fixed-share consists of applying AAD to the stochastic predictor that works as follows. It starts by choosing a static expert at random (with equal probabilities). On every trial it replicates the prediction of the chosen expert and then with probability  $\alpha$  switches to a different static expert choosing the target expert at random (with equal probabilities). Given that the initial distribution on the static experts is uniform, Proposition 4.5.1 implies the following bound on the loss of the AAD:

$$\text{Loss}_T(\text{AAD}) \leq c(\beta) \text{Loss}_T(E) + \frac{c(\beta)}{\ln(1/\beta)} \left( \ln N + k \ln(N-1) + k \ln \frac{1}{\alpha} + (T-k-1) \ln \frac{1}{1-\alpha} \right), \quad (4.9)$$

where  $N$  is the number of static experts and  $E$  is a dynamic expert making  $k$  switches. The derivation consists of evaluating the probability of making  $k$  switches (see [127], Section 3.3).

The variable-share algorithm consists of applying AAD to the stochastic predictor that works in the same fashion except that the probability of making a switch is taken to be  $1 - (1 - \alpha)^l$ , where  $l$  is the loss of the chosen expert on the latest trial. The probability of making a switch thus reduces if the chosen expert performs well ( $l$  is small) and increases if the chosen expert performs badly ( $l$  is large).

The analysis based on a direct application of Proposition 4.5.1 turns out to be rather crude; however a finer analysis (see [127], Theorem 3) yields the bound

$$\text{Loss}_T(\text{AAD}) \leq c(\beta) (\text{Loss}_T(E) + k) + \frac{c(\beta)}{\ln(1/\beta)} \left( \ln N + k \ln(N-1) + k \ln \frac{1}{\alpha} + \text{Loss}_T(E) \ln \frac{1}{1-\alpha} \right), \quad (4.10)$$

where the notation is as before and it is assumed that the initial distribution on the

static experts is uniform.

The practical application of fixed- and variable-share algorithms is greatly facilitated by the following observation. In order to apply (4.6), we only need to know the aggregate weights of the dynamic experts that follow each particular static expert.

Let  $w_t(\theta)$  be the aggregate weights assigned to the dynamic experts following a static expert  $\theta$ . We can initialise  $w_t(\theta)$  by an arbitrary distribution  $p_0(\theta)$ . On every trial the weights are updated as follows.

For the fixed-share algorithm we let

$$w_t^*(\theta) = w_{t-1}(\theta)\beta^{\lambda(\gamma_t(\theta), \omega_t)}$$

and

$$w_t(\theta) = (1 - \alpha)w_t^*(\theta) + \frac{\alpha}{|\Theta| - 1} \sum_{\tilde{\theta} \neq \theta} w_t^*(\tilde{\theta}) .$$

The former equation reflects the update of the weights caused by the loss; each dynamic expert following a particular static expert is subject to the same change of its weight. The later equation reflects the switching; the share equal to  $1 - \alpha$  of dynamic experts keep following  $\theta$ , while of the experts following each  $\tilde{\theta} \neq \theta$  the share equal to  $\alpha/(|\Theta| - 1)$  switches to  $\theta$ . Here  $|\Theta|$  denotes the size of the pool of static experts  $\Theta$ .

The later formula for the variable-share algorithm is as follows:

$$w_t(\theta) = (1 - \alpha)^{\lambda(\gamma_t(\theta), \omega_t)} w_t^*(\theta) + \frac{1}{|\Theta| - 1} \sum_{\tilde{\theta} \neq \theta} (1 - (1 - \alpha)^{\lambda(\gamma_t(\tilde{\theta}), \omega_t)}) w_t^*(\tilde{\theta}) .$$

#### 4.5.6 Tracking for Sleeping Experts

In this section we describe a combination of the aggregating algorithm for sleeping experts with the fixed- and variable-share algorithms. We will refer to this algorithm as AA-S.

The idea is as follows. We will be applying AA-S to merge a pool of dynamic experts

that can fall asleep. A dynamic expert falls asleep when it switches to a sleeping static expert.

As before, we maintain aggregate weights  $w_t(\theta)$  of dynamic experts following a static expert  $\theta$  on trial  $t$ . The weights are initialised by a uniform distribution  $p_0(\theta)$ .

On step  $t$  we output  $\gamma_t$  satisfying (4.5) with

$$g(\omega) = \log_\beta \frac{1}{\sum_{\theta \in \Theta_{\text{awake}}} w_{t-1}(\theta)} \sum_{\theta \in \Theta_{\text{awake}}} w_{t-1}(\theta) \beta^{\lambda(\gamma_t(\theta), \omega)} , \quad (4.11)$$

where  $\Theta_{\text{awake}}$  is the set of static experts awake on step  $t$ . Indeed, as explained in Subsect. 4.5.4, we can take sums over awake dynamic experts, which currently follow only awake static experts.

Let us write down update rules. While the sleeping experts do not appear in the formula for  $g(\omega)$ , they affect the update rules. The update proceeds as follows. First, let

$$w_t^*(\theta) = w_{t-1} \beta^{\lambda(\gamma_t(\theta), \omega_t)}$$

for every  $\theta \in \Theta_{\text{awake}}$  and

$$w_t^*(\theta) = w_{t-1} \beta^{\lambda(\gamma_t, \omega_t)/c(\beta)}$$

for  $\theta \in \Theta_{\text{sleep}}$ , the pool of static experts sleeping on step  $t$ ; in other terms, the weights of awake experts are updated with their losses and the weights of sleeping experts are updated with the learner's loss.

Secondly, we take into account the switches:

$$w_t(\theta) = (1 - \alpha) w_t^*(\theta) + \frac{\alpha}{|\Theta| - 1} \sum_{\tilde{\theta} \neq \theta} w_t^*(\tilde{\theta}) \quad (4.12)$$

for the fixed-share algorithm and

$$w_t(\theta) = (1 - \alpha)^{l(\theta)} w_t^*(\theta) + \frac{1}{|\Theta| - 1} \sum_{\tilde{\theta} \neq \theta} (1 - (1 - \alpha)^{l(\tilde{\theta})}) w_t^*(\tilde{\theta}) , \quad (4.13)$$

where

$$l(\theta) = \begin{cases} \lambda(\gamma_t(\theta), \omega_t), & \text{if } \theta \text{ is awake on step } t; \\ \lambda(\gamma_t, \omega_t)/c(\beta) & \text{otherwise} \end{cases} \quad (4.14)$$

for the variable share algorithm.

Note that all experts, both sleeping and awake, participate in switching and change their weights.

#### 4.5.7 Sleeping Specialist Experts Performance Bounds

Let  $E$  be a dynamic expert making  $k$  switches. Given that the initial distribution on the static experts is uniform, bounds (4.9) and (4.10) will hold for modified fixed- and variable-share, respectively. At times the static experts  $E$  switches to may fall asleep; those spans are excluded from the total losses of the sleeping experts algorithm and  $E$  because, as shown in Subsect. 4.5.4, the losses on the left- and right-hand sides cancel themselves out.

#### 4.5.8 Markov Chains of Experts

In prediction with expert advice the goal is to design on-line prediction algorithms that achieve small regret (additional loss on the whole data) compared to a reference scheme. In the simplest such scheme one compares to the loss of the best expert in hindsight. A more ambitious goal is to split the data into segments and compare to the best expert on each segment. This is appropriate if the nature of the data changes between segments. The standard fixed-share algorithm is fast and achieves small regret compared to this scheme.

The approaches of fixed- and variable-share may be generalised to cover Markov chains of experts [39].

In fixed- and variable-share the stochastic predictor assumes a uniform distribution on other experts when choosing an expert to switch to. This is not necessarily the best choice. We may have reasons to believe that a time span when an expert performs well

should always be followed by a timespan where the same other expert (or an expert from a fixed small subset) does well (in economic applications this amounts to assuming some temporal structure on the sequence of market regimes). Therefore some dynamic experts will have better chances of success over others; in view of Proposition 4.5.1 we may want to assign higher probabilities to them.

Let the stochastic predictor follow a Markov chain. Suppose we have transition probabilities  $p_{\tilde{\theta}\theta}$  of switching from expert  $\tilde{\theta}$  to  $\theta$ ; those may be constant or may change with time. The weight update rule will then be as follows. First, let

$$w_t^*(\theta) = w_{t-1} \beta^{\lambda(l(\theta), \omega)} ,$$

where  $l(\theta)$  is defined by (4.14) and, secondly, let

$$w_t(\theta) = \sum_{\tilde{\theta} \in \Theta} p_{\tilde{\theta}\theta} w_t^*(\tilde{\theta}) . \quad (4.15)$$

The fixed-share algorithm is a special case of this approach with transition probabilities

$$p_{\tilde{\theta}\theta} = \begin{cases} 1 - \alpha, & \text{if } \tilde{\theta} = \theta; \\ \frac{\alpha}{|\Theta| - 1} & \text{otherwise} \end{cases}$$

and the variable-share algorithm is a special case with transition probabilities

$$p_{\tilde{\theta}\theta} = \begin{cases} (1 - \alpha)^{l(\tilde{\theta})}, & \text{if } \tilde{\theta} = \theta; \\ \frac{1 - (1 - \alpha)^{l(\tilde{\theta})}}{|\Theta| - 1} & \text{otherwise} \end{cases}$$

(note that probabilities depend on the data via  $l(\theta)$ ).

In the general case we may consider arbitrary probabilities. The exact form of performance bounds analogous to (4.9) and (4.10) will vary and may be rather involved. However it should be sufficient for applications to rely on the intuition given by Propo-

sition 4.5.1: the transition probabilities should be chosen in such a way as to assign higher probabilities to dynamic experts that are believed to perform well.

## Chapter 5

# RTSSE Corpora

This chapter will explore the RTSSE corpora by use of visualization, and qualitative analysis of temporal structures to motivate forecasting algorithms.

### 5.1 Implied Volatility Prediction

In this section we describe a model problem in detail. Note that RTSSE is an abbreviation for the Russian Trading Stock System Exchange.

#### 5.1.1 Implied Volatility

An option is a derivative financial instrument linked to an underlying asset, which is usually a share, but can also be a portfolio of shares, a futures on a share etc. There are two popular types of options, puts and calls. Definitions and more details on puts and calls are available from standard textbooks such as [71, 133]; for the purposes of this thesis it suffices to note that puts and calls have two important parameters, strike price  $X$  and time to maturity  $T$ . Throughout the life of a particular option  $X$  stays fixed while  $T$  decreases and when it reaches 0, the option ceases to exist.

European and American options which differ by their execution arrangements. In this thesis we will deal with American options.

The most popular approach to options pricing is based on the Black-Scholes(-Merton) theory (see [15, 18, 32, 34, 96]). This theory assumes that the underlying asset price  $S$  follows an exponential Wiener process with constant volatility  $\sigma$ , which cannot be directly observed but can be estimated from historical data. Given  $\sigma$ , the prices of so called *European* call and put options, can be calculated using the Black-Scholes formulas

$$c = SN(d_1) - Xe^{-rT}N(d_2) , \quad (5.1)$$

$$p = Xe^{-rT}N(-d_2) - SN(-d_1) \quad (5.2)$$

with

$$d_1 = \frac{\ln(S/X) + (r + \sigma^2/2)T}{\sigma\sqrt{T}} ,$$

$$d_2 = \frac{\ln(S/X) + (r - \sigma^2/2)T}{\sigma\sqrt{T}} ,$$

where  $N$  is the cumulative distribution function of the Gaussian distribution with the mean of 0 and standard deviation of 1,  $S$  is the price of the underlying asset,  $\sigma$  is its volatility,  $r$  is the risk-free interest rate (assumed to be 0 in this thesis) and  $X$  and  $T$  are the strike price and time left until the maturity of the option.

In practice this model is often violated. The prices  $c$  and  $p$  can be observed directly in transactions as well as  $S$ . Given the current prices of options and the underlying asset we can find  $\sigma$  that satisfies the Black-Scholes equations. This  $\sigma$  is known as the implied volatility. Contrary to Black-Scholes theory, it is often not constant and exhibits a dependency on the strike price and time. The graph of a dependency of  $\sigma$  on the strike price  $X$  (other parameters being equal) is known as the volatility smile due to its characteristic shape; the graph of the dependency of  $\sigma$  on  $X$  and  $T$  is known as the volatility surface. There is no unique generally accepted theory explaining the phenomenon of implied volatility; however, volatility remains a meaningful parameter

and traders often use it to quote option prices. The reader may refer to Chapter 16 of [71] as a financial introduction to volatility smiles.

### 5.1.2 RTSSE Datasets

In this thesis we approach the problem of finding the implied volatility from a purely machine learning and time-series perspective. The datasets used were provided by the Russian Trading System Stock Exchange (RTSSE) and record data from mid-2000s, when the Russian stock market was experiencing steady unperturbed growth (perhaps unhealthy from an economics point of view).

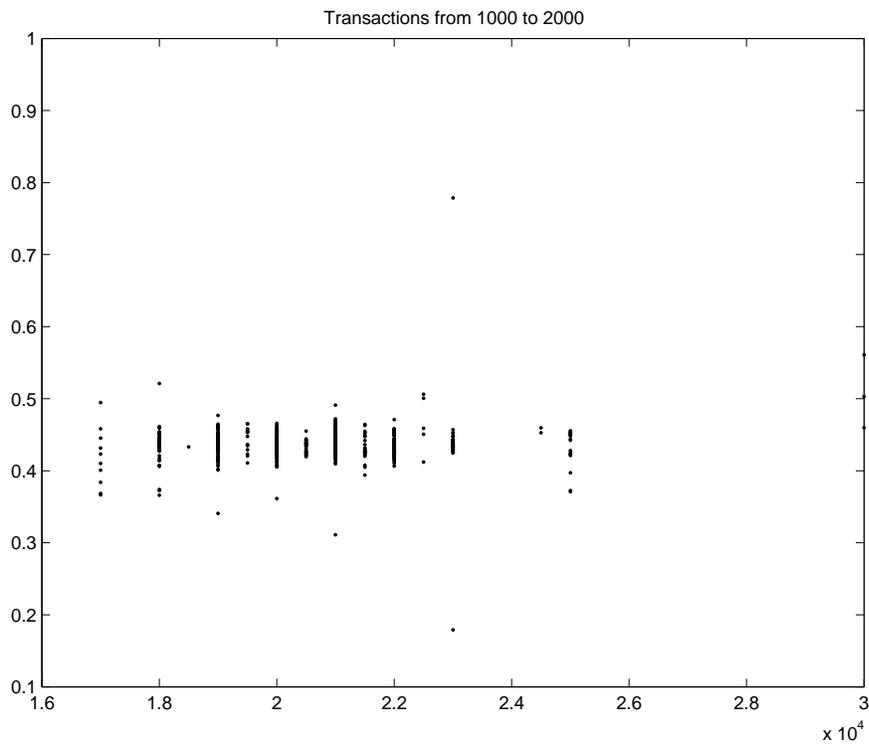


Figure 5.1: Volatility vs strike, transactions 1000-2000 on eeru1206.

The following three datasets were used: eeru1206 describes put and call options maturing in December 2006 on futures on shares of Unified Energy System of Russia (the company has since been split and its shares are no longer trading); gaz307 describes

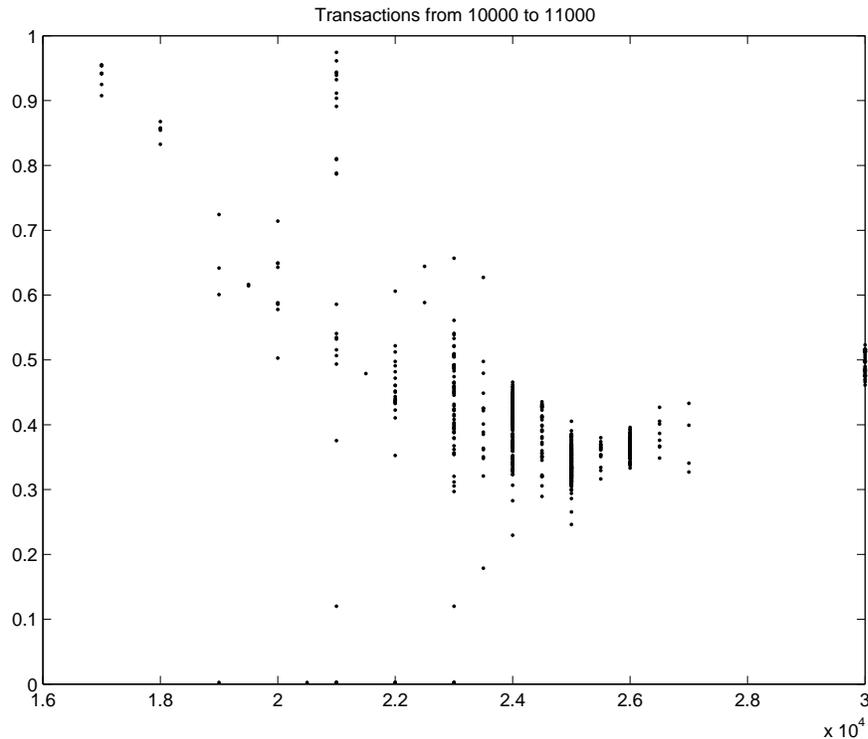


Figure 5.2: Volatility vs strike, transactions 10000-11000 on `eeru1206`.

put and call options maturing in March 2007 on futures on shares of Gazprom; and `rts307` describes put and call options maturing in March 2007 on futures on the RTSSE index.

The options studied were *American* rather than *European*, which implies a difference in the execution arrangements. Under very general assumptions, American call options should not be executed early and therefore they cost the same as European call options and should satisfy (5.1). The same cannot be said of American put options, which can cost more than European put options. Formula (5.2) is technically speaking not applicable to American put options. However one can still calculate implied volatility using the Black-Scholes formula; it has no meaning within the standard Black-Scholes model, but it was used by the RTSSE as a descriptive parameter.

Each dataset contains records of consecutive transactions with put and call options

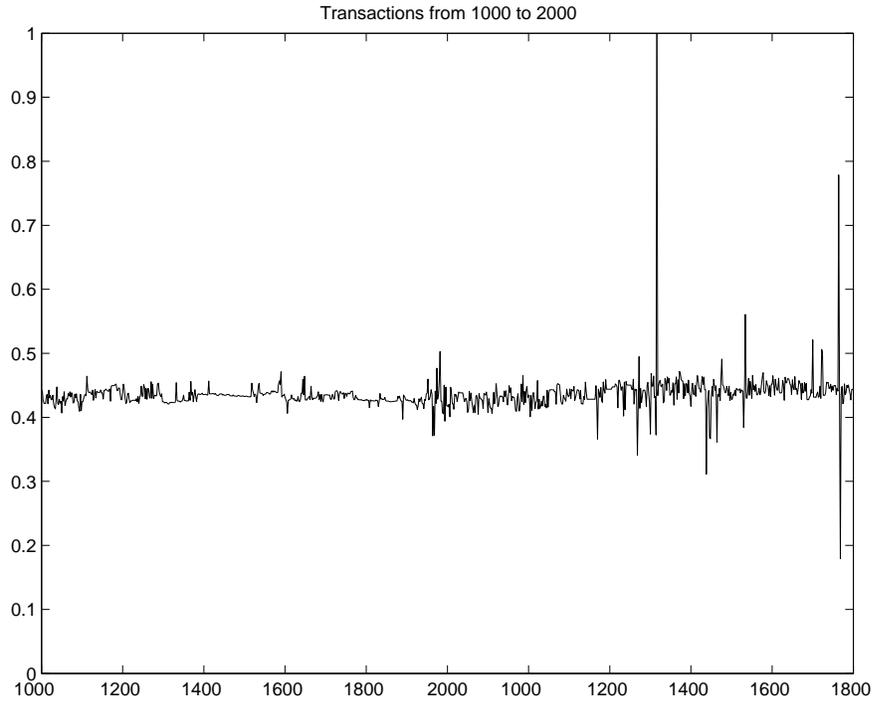


Figure 5.3: Volatility vs number, transactions 1000-2000 on `eeru1206`.

on a particular underlying asset with the same maturity date. The numbers of transactions are given in Table 5.1. The following attributes of a transaction are available: date and time, strike price  $X$ , time left to maturity of the option  $T$ , the price of the underlying asset  $S$  at the time of transaction, and a bit differentiating puts from calls. We attempt to predict the implied volatility in the transaction. The quality of the prediction is measured by the squared deviation of the predicted implied volatility from the true implied volatility calculated from the option price using the Black-Scholes formula (note that the option price is not one of our attributes; otherwise the problem would amount to learning the Black-Scholes formula itself).

The squared deviation is useful because;

1. Squared deviation is standard in statistics.
2. Square loss is mixable and therefore convenient from the point of view of prediction with expert advice (see [24, 28, 128, 130]).

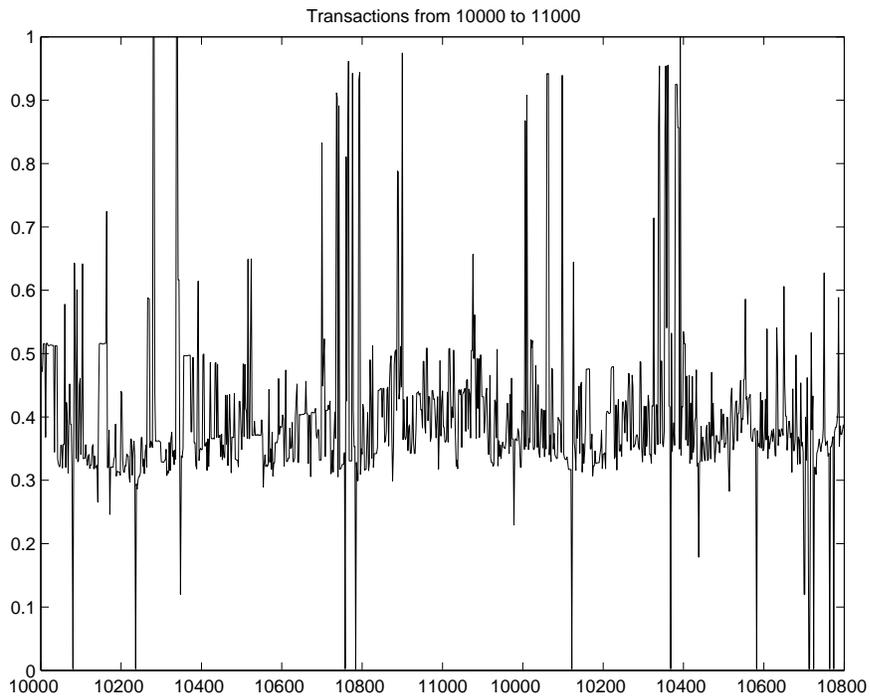


Figure 5.4: Volatility vs number, transactions 10000-11000 on `eeru1206`.

3. The stock exchange was interested in the problem of predicting volatility because they needed to produce quotes. Later people would use the quotes in their trading strategies for hedging or speculation. We do not know how exactly they would use the values and what effect deviations will make – this depends on a particular strategy. So the square loss is a reasonable compromise.

5. People normally use (polynomial) regression for modelling implied volatility (see the paper I quote) and regression is usually studied in the context of square loss.

6. It will guarantee a positive number and also punish increasing losses more severely.

The losses incurred by our methods are compared against a benchmark technique employed by the stock exchange for quoting implied volatilities. This is a proprietary method based on fitting coefficients in a formula describing the volatility smile; the method involves occasional manual adjustments. As the outputs of the proprietary

Table 5.1: Datasets Summary

Dataset	Underlying asset	Maturity	Number of transactions
<code>eeru1206</code>	futures on share	December 2006	13152
<code>gaz307</code>	futures on share	March 2007	10985
<code>rts307</code>	futures on index	March 2007	8410
<code>rts1206</code>	futures on index	December 2006	10126

technique were used for publicly available quotes, one cannot rule out the influence of the technique on the behaviour of the implied volatility, which makes competing with this method particularly difficult.

Note that `rts1206` will be used as a validation corpus.

## 5.2 Qualitative Analysis

### 5.2.1 Time Progression

Refer to Figure 5.5 for an illustration of the progression of time in relation to record number. The corpora contains only options nearing maturity and time progression is close to linear although could be said to slow down near maturity. In other words; there are slightly less transactions near maturity. However; as time is close to linear. When strikes and volatility are plotted against record number in the coming sections; time and record number will be regarded as synonymous.

### 5.2.2 Implied volatility Distribution

See Figures 5.13 (`rts307`), 5.14 (`gaz307`) and 5.12 (`eeru1206`) for visualizations of the volatility (label) distribution against time in the corpora.

In `eeru1206` the label distribution is more changeable at the beginning which is in stark contrast to the other corpora. The labels begin around 0.5 on average with minimal variability and then slowly trend down to 0.35. Throughout the corpus, the

labels are distributed slightly wider than the other corpora with the following caveat. It is more regular and flat at the end. `gaz307` and `rts307` have a characteristic *bend* in the average values near maturity and are generally far more similar to each other than either are to `eeru1206`.

The first 4 to 6 thousand records are highly regular for `rts307` and `gaz307`. For example; in the case of `rts307` a predictor which only predicted 0.4 would presumably suffer minimal loss. Therefore. Close attention should be given to the performance of any predictor at the end of the corpora, because the nature of the data changes near maturity and becomes increasingly chaotic. It is clear from these Figures that volatility is harder to predict well near maturity.

### 5.2.3 Strike Distribution

See Figures 6.7 (`eeru1206`), 6.6 (`rts307`) and 6.5 (`gaz307`) for a visualization similar to a spectrogram in music analysis of how strikes are distributed in respect of time. Note that strike rows are only shown if records with that strike are present in the corpora.

The movement of the strikes before maturity tells an interesting story. On `gaz307`; the strikes average 30000 at the beginning of the corpus and shifts slowly down towards 26000 on average near maturity (with some minor acceleration to note towards the end). The strikes range from 22000 to 40000 and move up in increments of 1000.

On `rts307` the picture is more chaotic. The strikes appear to be shifting somewhat erratically with an early average of 185000. After record 4500 an 'S' bend is clear. The strikes range from 140000 to 220000 and move up in increments of 10000 and 5000.

On `eeru1206` the picture is quite regular. The strikes appear to be shifting upwards towards 26000 with an early average of 21000. The strikes range from 16000 to 30000 and move up in increments of and 500.

### 5.2.4 Self-Similarity Map With Predictive Regions

Following from the work on discovering temporal structures in music (see Chapter 3); the self-similarity structure of the RTSSE corpora is visualized using a 2-dimensional image map, constructed as follows.

$R$  equally-sized ridge regression models, equally-spaced throughout the corpora were trained and validated on themselves. The training/validation window size  $w$  used was 200. The number of models  $R$  was 400. The models were positioned from  $\max(c - w + 1, 1)$  to  $c$  for each  $c$  in

$$c_i = \left\{ 1 \frac{N - w + 1}{R}, 2 \frac{N - w + 1}{R}, \dots, R \frac{N - w + 1}{R} \right\}$$

Where  $N$  is the number of examples in the corpus.

Vapnik's polynomial kernel was used and the validation procedure for each region was a brute-force search across a suitable range of parameters; degree  $d \in \{1, 2, \dots, 7\}$  and ridge coefficient

$$a \in \{0.2, 0.4, 0.6, \dots, 10\}.$$

A self-similarity  $N \times R$  matrix is constructed showing the log-loss (a non-linear scale is required to see the nuance in the temporal structures as there are a high dynamic range of variations in value and a simple colour interpolation would not have been sufficient to show details).

Refer to the following subsection for discussion of the Figures produced by this algorithm.

## 5.3 Temporal Structures Discussed

See Figures 5.6 (`eeru1206`), 5.7 (`gaz307`), 5.8 (`rts307`) for visualizations of the self-similarity structure of the corpora as discussed in the previous subsection.

And see Figures 5.9 (`eeru1206`), 5.10 (`gaz307`), 5.11 (`rts307`) for visualizations

of the self-similarity structure of the corpora that have been annotated with temporal structures.

The following temporal structures are immediately apparent from these illustrations.

- Self-similar regimes.
- Regions that evolve dynamically are not similar to anything else.
- Hard and soft regime boundaries.

In *eeru1206*) temporal structures are less apparent than the other corpora. There is a strongly self-similar region near the beginning of the corpus (A). Regimes B, C and D have been annotated but are transitory and have soft boundaries. The clearest change in regime/region is between the high level (blue coloured) regions marked E and F. It means that few records in regime F would be able to convey any predictive information about regime E. *However, records in (green) regime M would be able to convey predictive information about the entire corpus, performing the worst on regime D.* This is one of the key tenets of this thesis and raises the following comments. We will introduce the term *master key temporal structure* to refer to structures that appear to have a highly descriptive nature across the corpus in question.

- If the nature of corpora changes as a function of time, validation across the entire corpus no longer makes sense if globally minimized loss is required.
- If validation needed to be economical in some sense, would regime M be the best region to validate on?
- Is there a trade-off by validating on regime M? Perhaps spatial dependencies are emphasised because the region contains more information and therefore requires an increasingly complex model.

- It is clear that recent information nearly always provided information about its future vicinity. This is the premise of on-line sliding window forecasting algorithms. But could old information elucidate about a regime in the future? Perhaps during its introductory phase or just in general.

The regime marked D which represents the time nearing maturity is still somewhat similar to region C, which is why the extension region C2+C+D is marked.

In `gaz307`) the following observations can be made.

- The period of time leading up to maturity (region E) is highly unpredictable. It can not even predict itself. Other regimes in the corpus convey no information whatsoever about regime E.
- It would be possible to conflate A and the first half of B with a few gaps of similarity.
- Regimes A, B and C are strongly self-similar. Regions A and C are not similar apart from a few records in A that seem to describe B and C well (marked in blue).
- There are no obvious *master key regions*.
- B is a dynamically evolving, transitory region. The beginning does not predict the end well.
- C is a region with near contiguous self-similarity like the music tracks in the corpora in Chapter 3.

Some comments on `rts307`.

- A and B are the two obvious regimes of note in this corpus.
- A and B could be conflated but there is an interval in A which doesn't predict B well at all (500-1300).

- B is a dynamically evolving, transitory region. The beginning does not predict the end well.
- The region near maturity C is weakly self-predictable, and is slightly predictable from records 500 onwards.

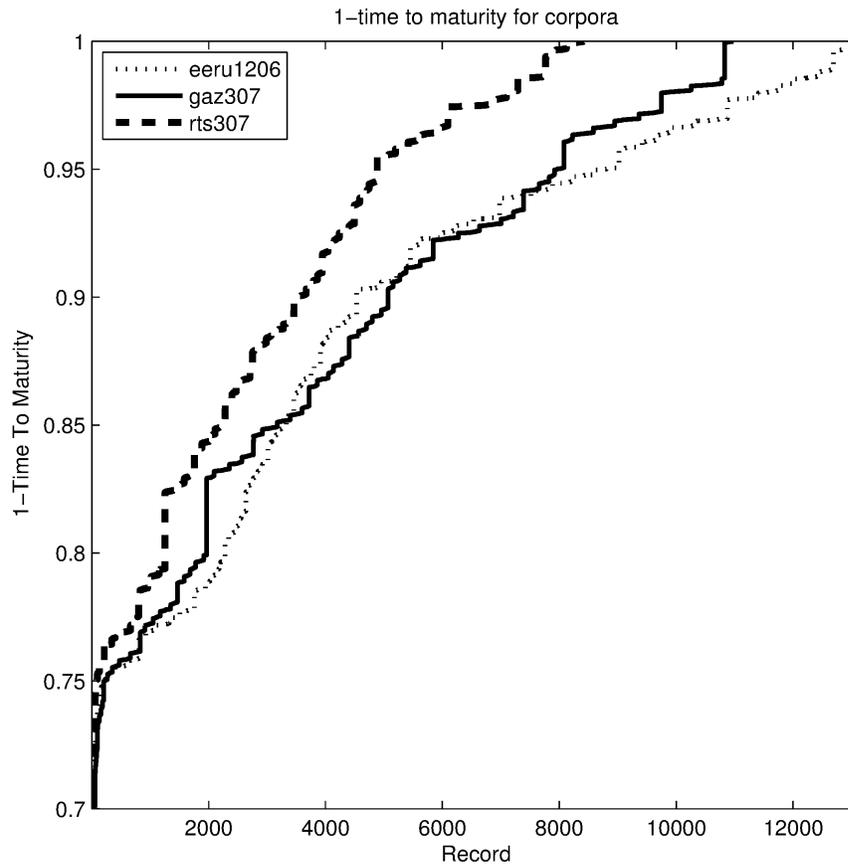


Figure 5.5: Plots of the time to maturity signal on the corpora. The corpora contains only options nearing maturity and time is close to linear.

## 5.4 Exploiting Temporal Structures

Ridge Regression and similar algorithms can suffer from a common problem. To apply their kernelized versions in an on-line fashion, one needs to deal with a kernel matrix of

size  $t \times t$  (or similar), where  $t$  is the current time. Standard methods such as Cholesky decomposition can be adapted to achieve a linear time incremental update (although they can still accumulate some numerical error). There is no obvious argument citing time -complexity or space-complexity to use anything other than the sliding window kernel ridge regression. Could there be a domain-specific or abstract reason?

The self similarity images discussed in Section 5.2.4 are intriguing and many of the temporal structures observed were reminiscent of those in the music corpora.

On-line algorithms invariably deal with a rectangular sliding window which retrieves temporal or spatial relationships in the immediate past whether in time-series forecasting [90] or regression [20, 21].

Is a recent sliding (or jerking) window the best way to achieve the lowest cumulative loss in an on-line forecasting algorithm? Or could the observed temporal structures be incorporated into an on-line algorithm?

Using a specialist expert algorithm would allow results to be obtained that were as close as possible to the best expert sequence without foreknowledge of the temporal structure of the regions.

Figure 5.15 shows the same self-similarity image map as described in subsection 5.2.4, for the `eeru1206` corpus. However rather than having the squared deviation of error for individual records shown, they have been grouped into adjacent tiles of the nearest integer to size

$$\frac{N}{110}.$$

A slightly modified version of the music segmentation algorithm described in Chapter 3 is then executed on the matrix. It is attempting to find the least cost path through matrix given a fixed number of switches (which would be the same as tracks in the music algorithm). Recall that each cell in the matrix is the sum squared deviation of another region experts making predictions on all records in the region.

The algorithm for discovering these temporal structures is available on GitHub <sup>1</sup>.

<sup>1</sup>[github.com/ecsplendid/MergingRegionExpertEvaluators/blob/master/map\\_losspath.m](https://github.com/ecsplendid/MergingRegionExpertEvaluators/blob/master/map_losspath.m)

Like the algorithm scribed in detail in Section 3.5, it uses the dynamic programming trick that the solution for a fixed number of switches at a certain point in time, builds on the solution for the number of switches minus 1.

The Figure shows the minimum cost path for 4, 8, 16 and 32 switches respectively. The encouraging result was that when 32 switches were selected, rather than always using newer models (such is the case in almost all on-line regression algorithms) at 3 points in the path it switched *backwards* to consult *older information*.

This is the key idea that is going to be developed in the following chapter when various new approaches to on-line forecasting are considered.

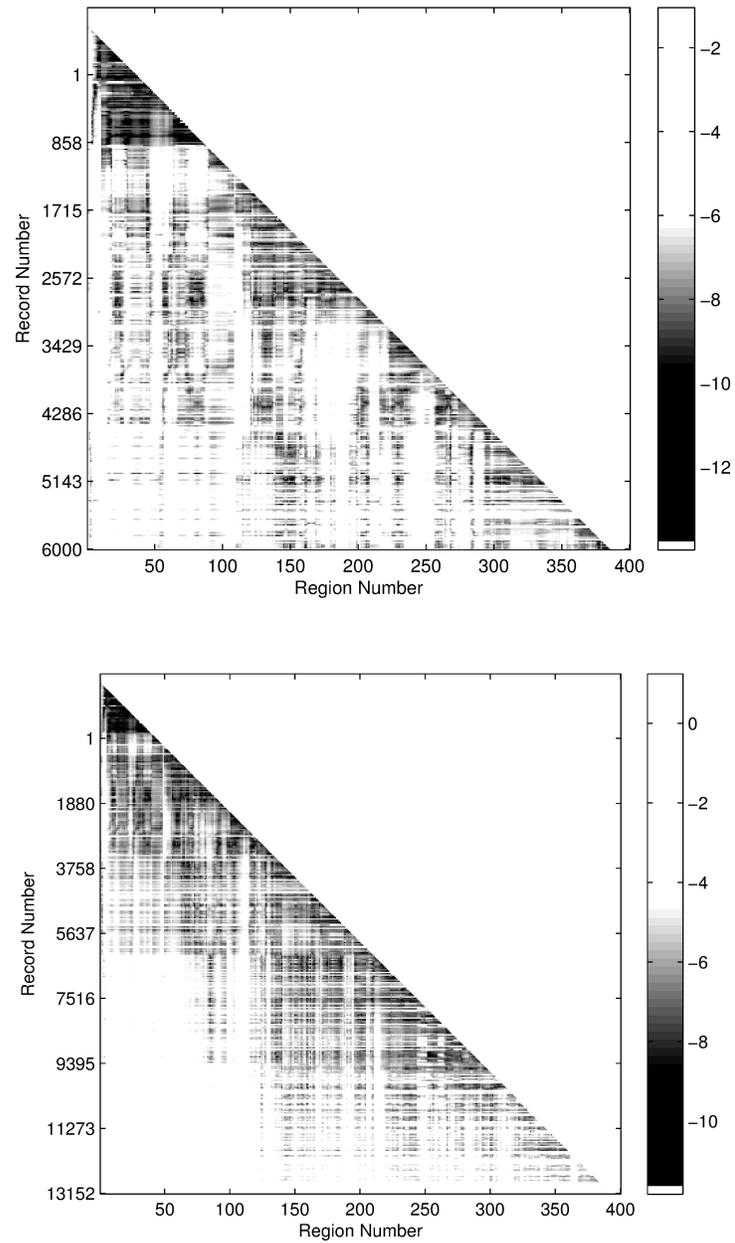


Figure 5.6: eeu1206 self-similarity/region analysis. Global analysis (top), first 6000 records (bottom).

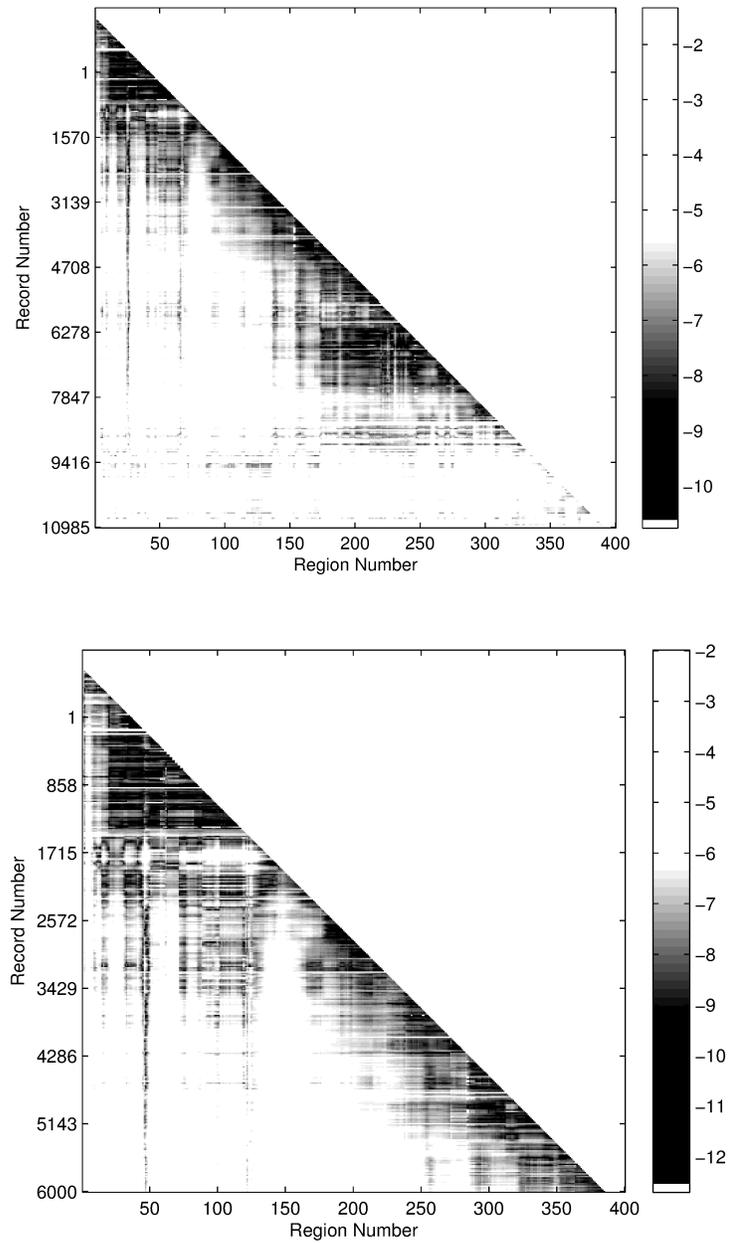


Figure 5.7: gaz307 self-similarity/region analysis. Global analysis (top), first 6000 records (bottom). The first 6000 records show two distinct regions with a transitory overlapping section.

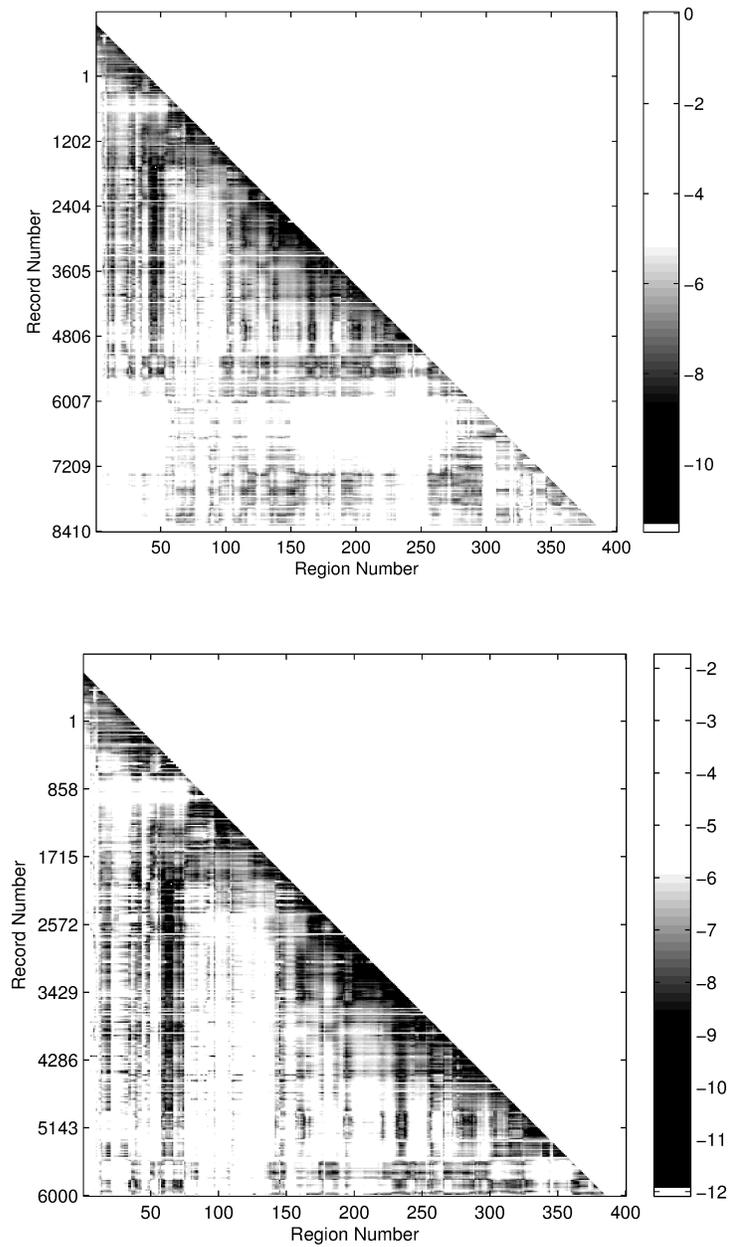


Figure 5.8: rts307 self-similarity/region analysis. Global analysis (top), first 6000 records (bottom). Perhaps the most interesting data set from our perspectives. The first 6000 records show alternating self similar regions.

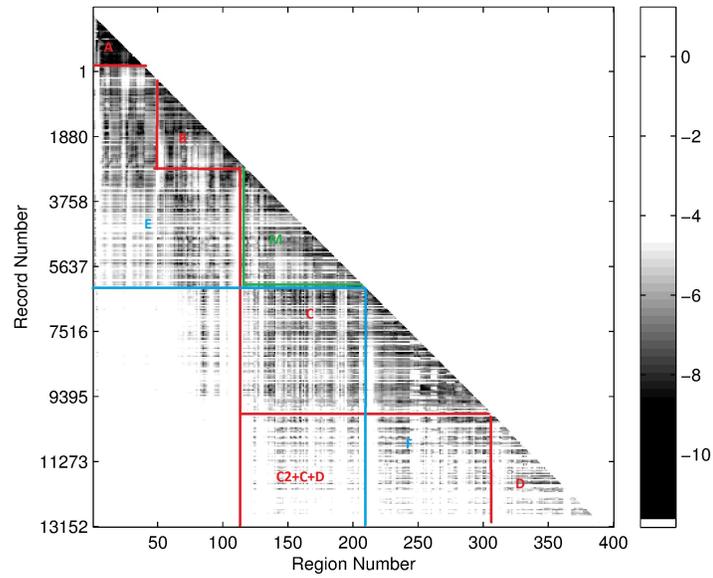


Figure 5.9: eeru1206 self-similarity and analysis of temporal structures.

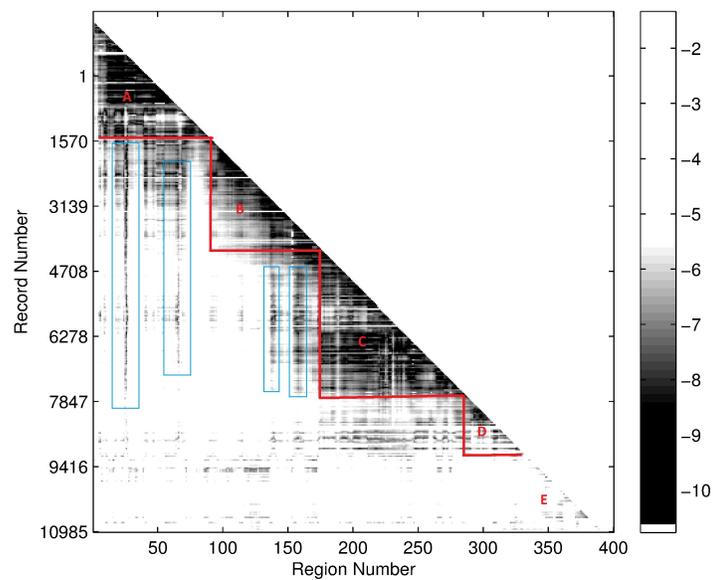


Figure 5.10: gaz307 self-similarity and analysis of temporal structures.

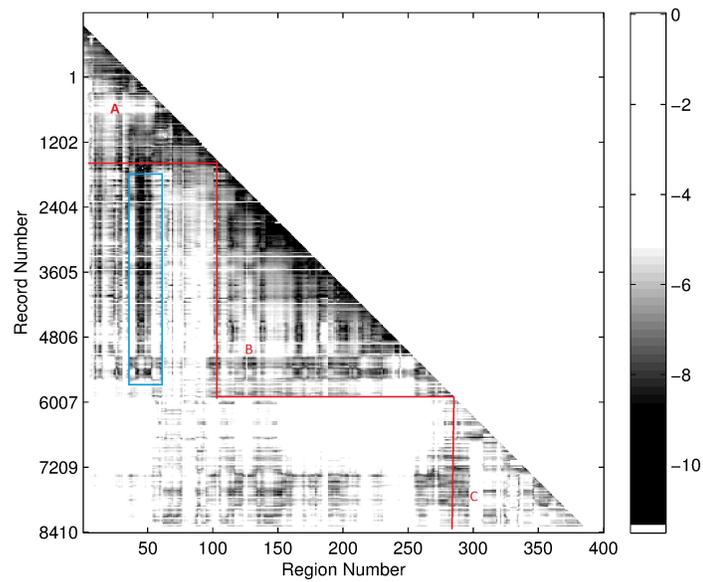


Figure 5.11: rts307 self-similarity and analysis of temporal structures.

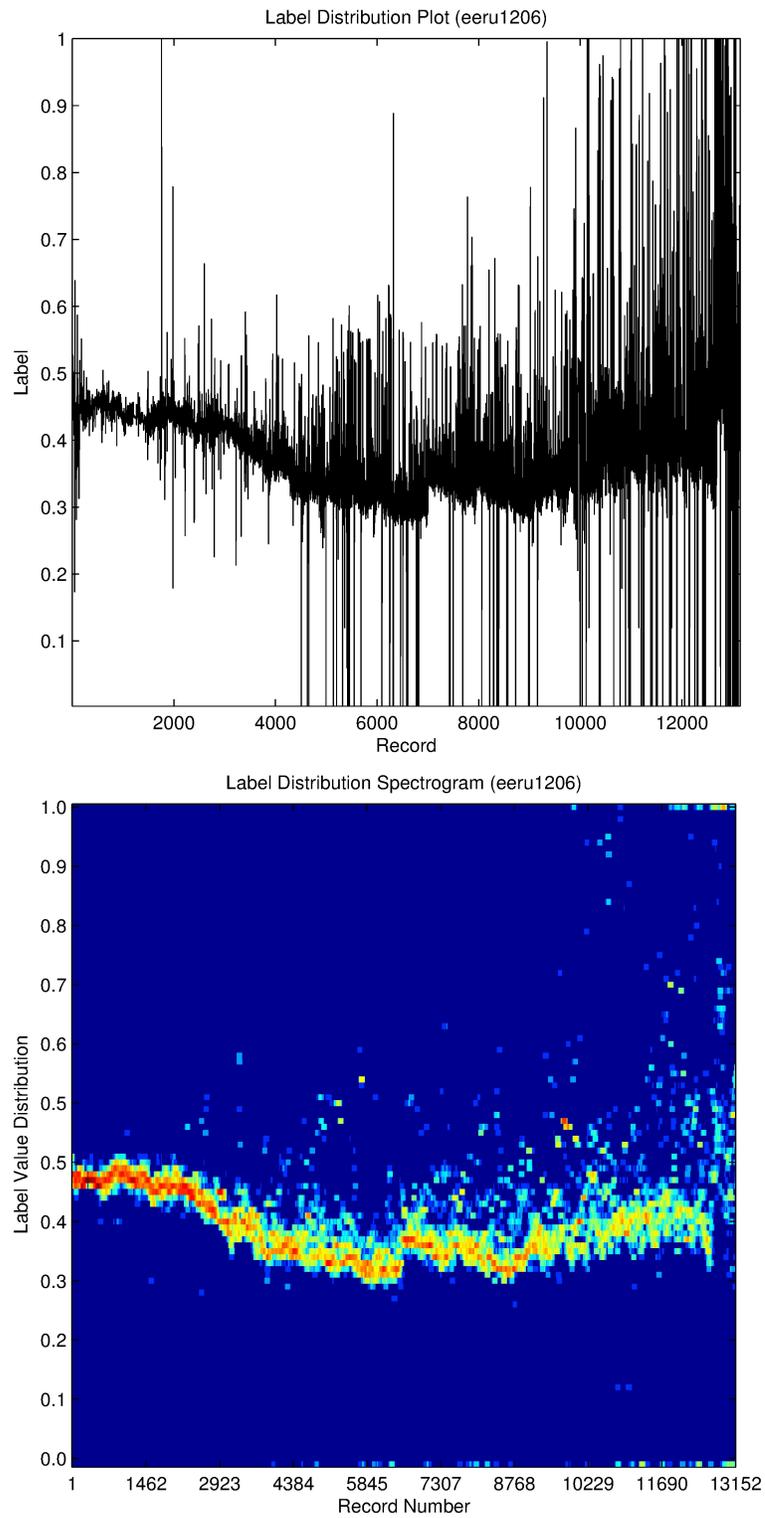
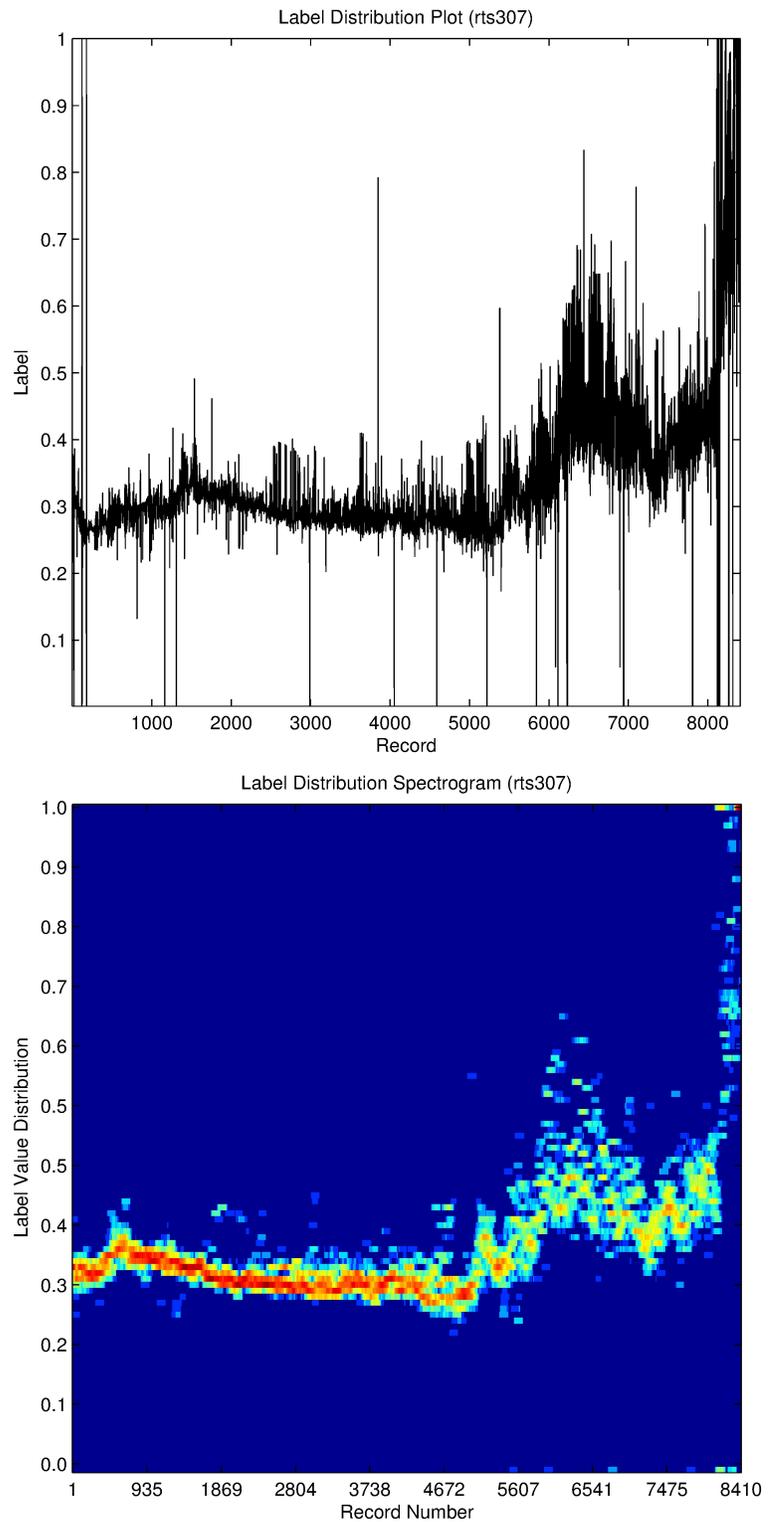


Figure 5.12: Volatility (label) Distribution on eeu1206.

Figure 5.13: Volatility (label) Distribution on `rts307`.

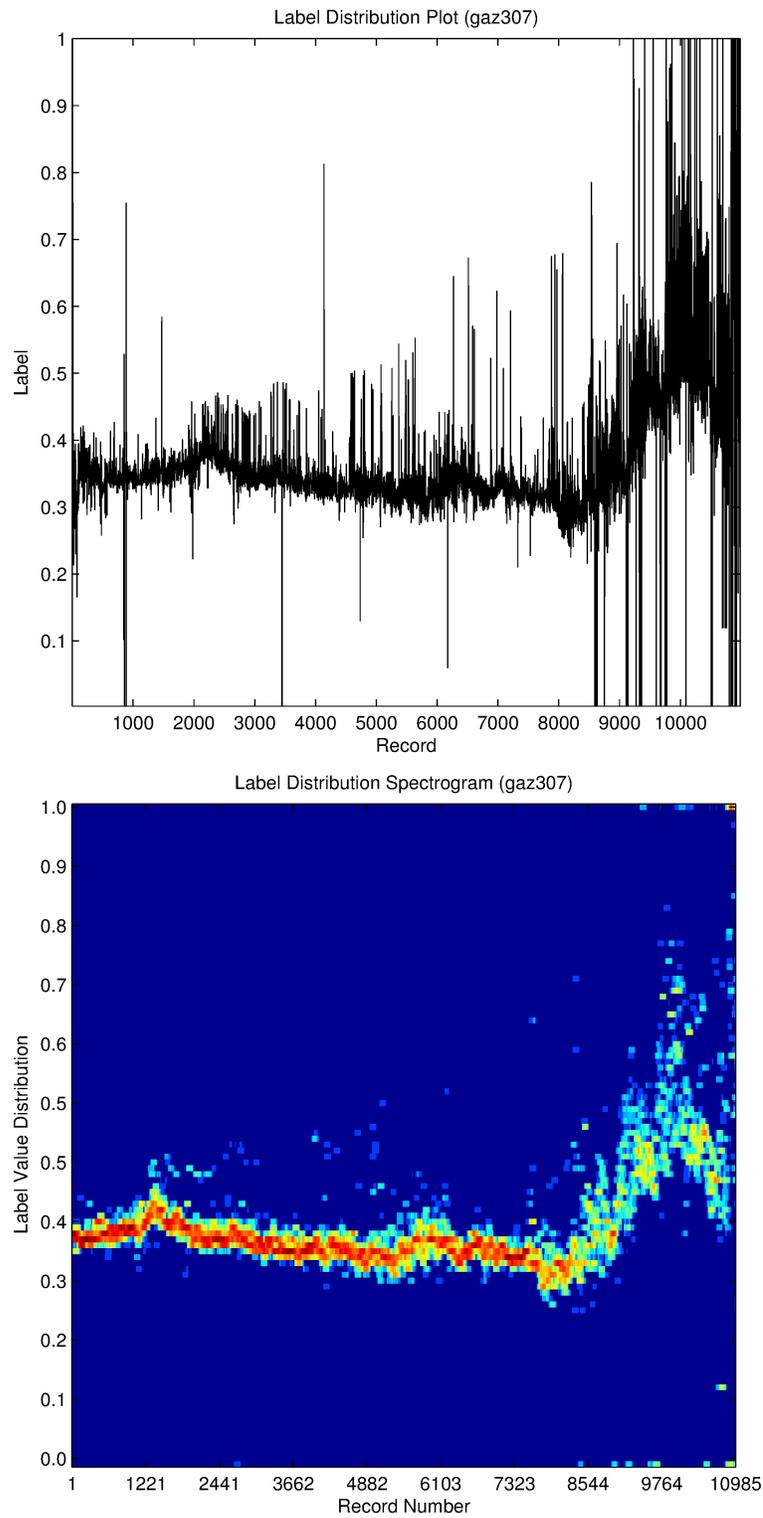


Figure 5.14: Volatility (label) Distribution on gaz307.

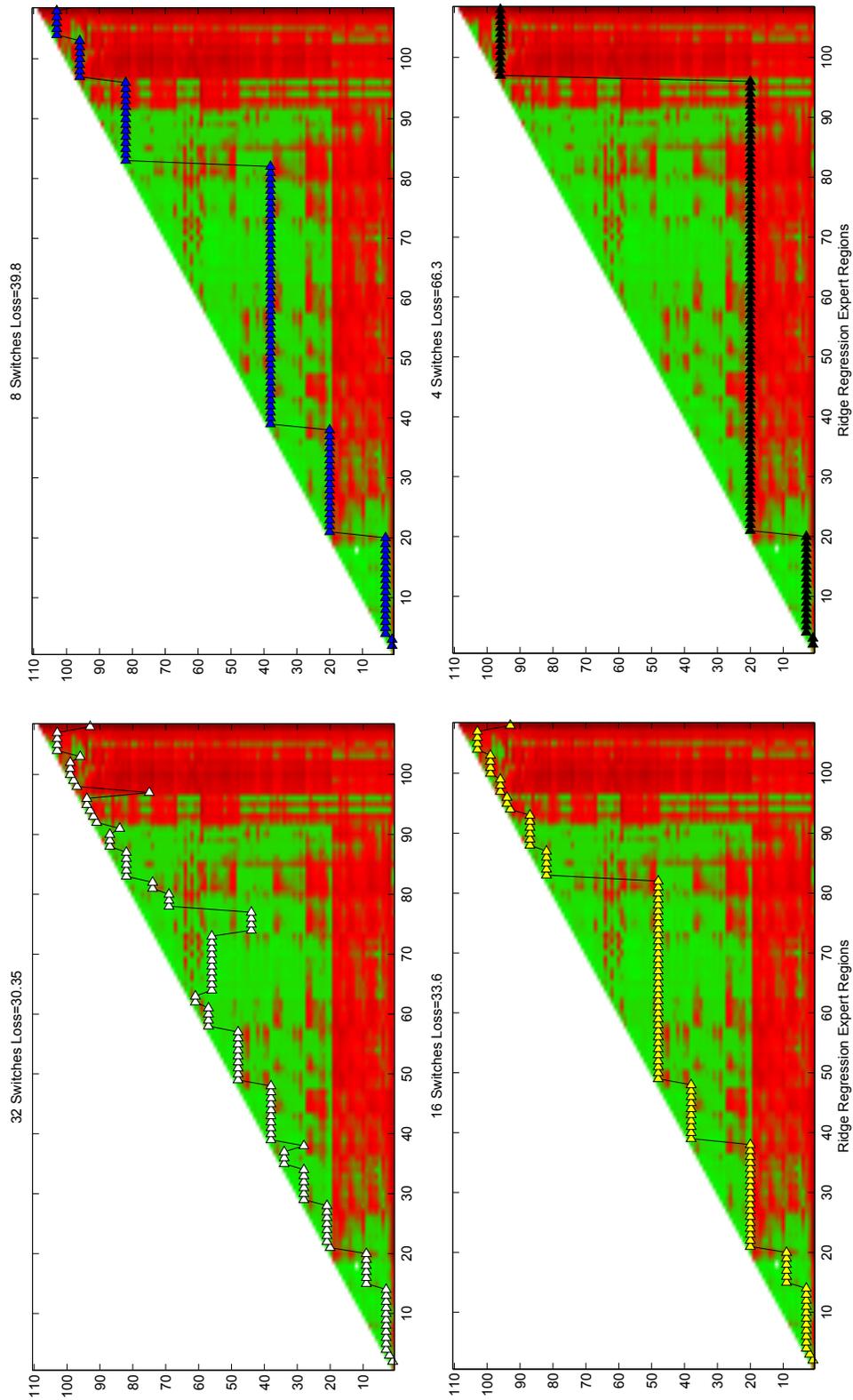


Figure 5.15: The concept for a region experts algorithm.

## Chapter 6

# On-line Forecasting With Specialist Experts

This chapter will introduce some new algorithms that implement specialist techniques to on-line forecasting.

### 6.0.1 On-line Kernel Ridge Regression

In this section we report results of kernel ridge regression for comparison. Ridge regression is a popular method of machine learning (see Subsect. 4.2) and it can be seen as an extension of model building used in econometrics.

Kernel ridge regression is the most natural competitor for any on-line forecasting algorithm that may be introduced as it is capable of modelling spatial dependencies from the side domain (the input signals) and it is a tried-and-tested method that is used extensively in the literature and industry (for example; [4, 22, 108, 116, 129]).

A genetic algorithm was executed on the validation set `rts1206` to find the best parameters for kernel ridge regression. These parameters were as follows. The ridge coefficient

$$a \in [0.0001, 10],$$

Vapnik's polynomial kernel degree

$$d \in \{1, 2, \dots, 10\}$$

and the sliding window size

$$w \in \{20, \dots, 450\}.$$

The stochastic optimization genetic algorithm `ga`<sup>1</sup> function built into MATLAB was used to find these parameters. Genetic searches are a robust way of finding optimal or near-optimal parameters on a multitude of optimization problems, which can be non-convex and have integer constraints. The genetic algorithm approach to parameter searching has been proven in the literature [122]. For more details on the method see [55].

The parameters used for the genetic search (and for all the genetic searches used in the following subsections) are as follows.

The population size is 40. This is how many random instances get initially generated (continuously distributed across the allowable space). Mutations are then generated using a Gaussian random function and the variance of which shrinks as function of generations passed. This means that near the end of the algorithm the variability will be reduced (this is a similar concept to simulated annealing [1]).

The elite count is 5 which means that the best 5 instances survive to the next generation.

The crossover fraction used is 0.6 which means 60% of the new generation are a result of crossovers of the best of the last generation and the remaining 40% are random mutations of the best of the last generation.

The stopping criteria used is 5 generations or a *stall limit* of 4. A stall limit means when the best solution does not change for a fixed number of generations.

The optimization function was to minimize sliding kernel ridge regression on valida-

---

<sup>1</sup><http://uk.mathworks.com/help/gads/ga.html>

tion corpus `rts1207` and the median average sum squared deviation of implied volatility was the evaluation score.

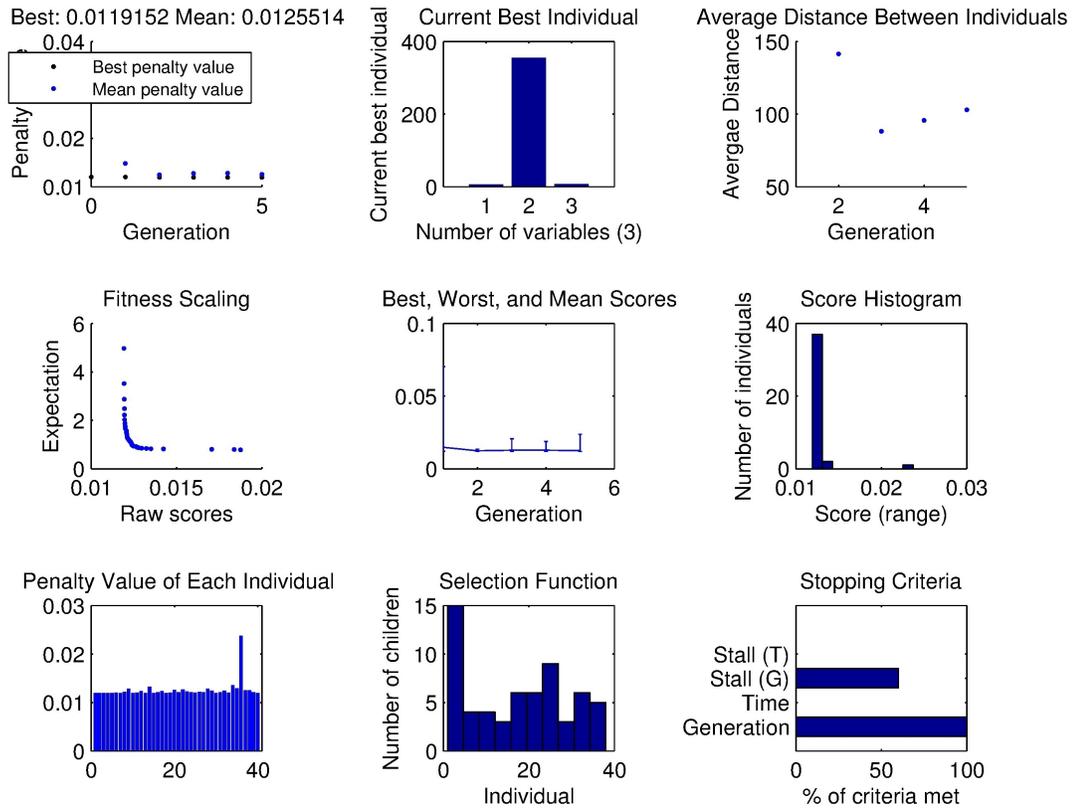


Figure 6.1: The graphical output of MATLAB for the `ga` (genetic algorithm) routine, finding the best parameters for the on-line kernel ridge regression procedure.

See Figure 6.1 for the output of MATLAB while searching for these parameters. The Figure is highly informative. For example; it shows in the top left the average and best penalty function for each generation.

The optimal parameters found were

$$a = 0.1695$$

Vapnik's polynomial kernel degree

$$d = 4$$

Table 6.1: Cumulative ridge regression loss at the end of the dataset, when validated on corpus `rts1206`.

Corpus	Improvement achieved by RR	Parameters		
		$d$	$a$	$w$
<code>eeru1206</code>	34.04	7	7.7	355
<code>gaz307</code>	2.78	7	7.7	355
<code>rts307</code>	36.81	7	7.7	355

and the sliding window size

$$w = 355.$$

As input, ridge regression was given all parameters of the transaction enumerated in Subsect. 5.1.2: strike price, put/call bit, the price of the underlying asset, and the time left to maturity; each parameter was normalised to fit the interval  $[-1, 1]$ . Vapnik’s polynomial kernel of degree  $d$ , i.e.,  $\mathcal{K}(x_1, x_2) = (\langle x_1, x_2 \rangle + 1)^d$  was used. Regression was applied in a sliding window fashion with the window size of 355. On step  $t$  regression was trained on the training set formed by 355 previous pairs  $(x_{t-356}, y_{t-356}), (x_{t-355}, y_{t-355}), \dots, (x_{t-1}, y_{t-1})$ ; regression was thus retrained on every step. For the first 355 transactions prediction was done using the volatility from the previous transaction (recall from Figure 5.3 that at the beginning of the dataset volatility is very stable). The regression algorithm is not particularly sensible to window size. During testing, we found that values from 40 upwards performed well and the parameter was convex. On the RTSSE corpora, sliding kernel ridge regression performs much better near maturity if its models are self-validated. This is due to the chaotic and unique nature of the data at that time. See Table 6.1 for the parameters that were validated on `rts1206` and Table 6.2 for the parameters that were self-validated. Because it would be cheating to self-validate kernel ridge regression, the extrinsically-validated parameters were used.

Table 6.1 shows the figures of adjusted loss at the end of the dataset for the retro-

Table 6.2: Cumulative ridge regression loss at the end of the dataset, when self-validated with a fixed window size.

Corpus	Improvement achieved by RR	Parameters		
		$d$	$a$	$w$
eeru1206	26.82	4	1	250
gaz307	4.09	4	0.4	250
rts307	94.80	5	0.2	250

spectively optimal parameters, degree  $d$  and ridge  $a$ .

Figures 6.2–6.4 show the cumulative adjusted loss of regression for comparison with our methods. The pictures were plotted to the same scales as those in Figures 6.8–6.13; for Figure 6.4 this required cutting off parts of the graph.

See Figures 6.2, 6.3 and 6.4 for a plot of the cumulative adjusted squared deviation on the corpora.

The best-found parameters for this and all the following algorithms are displayed in Table 6.5.

## 6.1 Merging Temporal Dependencies

In this section we describe an algorithm for a predictor working in the environment of Protocol 3 (see Section 4.5.1).

The behaviour of volatility is illustrated in Figures 5.1–5.4, which show dependencies of volatility on strike and transaction number (which effectively means time) near the beginning and near the end of eeru1206. Initially volatility is relatively flat. Towards the end of the dataset it becomes much more variable and the dependency of volatility on strike takes on a smile-like shape. The range of the plots was capped at 1, so occasional outliers exceeding 1 are not shown. It is clear from the visualizations in Section 5.2.2 that any values above 1 are outliers.

The figures show that on the one hand, the sequence of volatilities looks very much

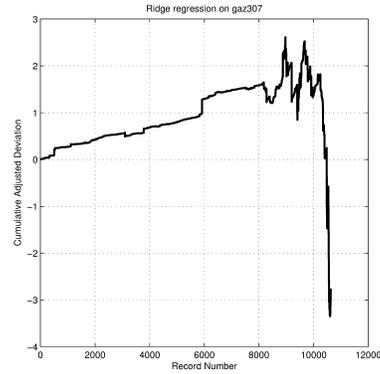
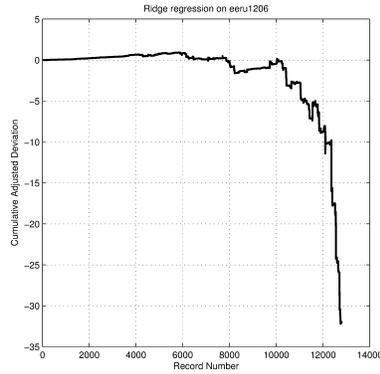


Figure 6.2: Ridge regression on eeru1206

Figure 6.3: Ridge regression on gaz307

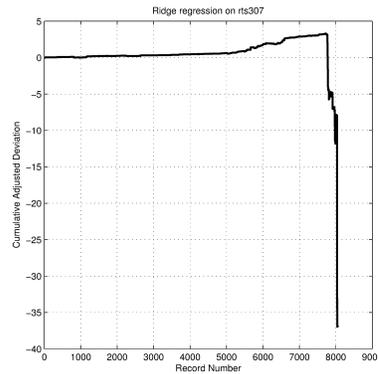


Figure 6.4: Ridge regression on rts307

like a time series and on the other hand volatility exhibits a dependency on strike price, especially towards the end of the dataset (i.e., close to maturity of the option).

See Figures 6.5, 6.6 and 6.7 for an visualization of the layout of the data in the strike-time domain. The strike-time dependency is rather clear and when viewed in this domain, the clusters of strikes appear as homogeneous temporal structures.

These figures provide the basis for our assumption that a record which is in the strike vicinity but closer in time might be better than a record which lacks recency but is the correct strike.

The number of possible strikes is limited. While theoretically the strike can have any real value, stock exchanges usually restrict strikes to some round numbers in order to improve liquidity. Thus one can consider splitting the time series into separate time

series, one for each strike. However it is apparent from the figures that some strikes are much more common than others. While for popular strikes the most recent transaction may be quite near, for rare strikes it is far away in time because there are fewer of them (see Figure 6.5).

Note that we cannot straightforwardly consider this problem within the framework of multivariate time series as, e.g., in [91]: we cannot speak of an evolution of a vector of volatilities for different strikes because of an irregular nature of the sequence of strikes. As a matter of fact, most standard models for volatility smiles are based on vectors of closing prices as in, e.g., [81].

One thus is naturally drawn to the idea of grouping strikes together and forming combined time series for groups of strikes. However the figures do not seem to suggest a natural grouping. It is known that most transactions happen for strikes near the current underlying price and the strikes on the sides are less frequently used. While the figures seem to be in agreement with this generally, there is no simple pattern in the distribution of strikes.

So one idea is born. Let us create multiple vicinities of strikes, introduce associated specialist experts, and let the sleeping experts algorithm converge on the relevant ones.

We are attempting to approach on-line prediction combining spatial dependencies with temporal evolution based on the use of specialist experts. In current practice, forecasters in macroeconomics and finance face a trade-off between model complexity and forecast accuracy.

So far, our methods have depended on parameter selection tuned on the domain beforehand. Also our methods did not directly incorporate domain specific temporal structures; instead relying on the protocol and kernel function to find structures that we may or may not have been aware existed.

Let us re-approach the problem by modularising it based on vicinities of data in the side domain. More parsimonious predictors can then be used and we can rely on the AAS (sleeping algorithm) to do the on-line learning.

Signals  $x$  are grouped into a finite number of vicinities and time series of outcomes  $y_t$  are formed according to what vicinities  $x_t$  belong to. The predictions made for time series are then merged using a prediction with expert advice technique.

This section will focus on the recent specialist expert techniques developed in [26, 27] (see Subsect. 4.5.4) to handle the vicinities with greater flexibility.

This problem falls in a somewhat grey area between the theory of time series and machine learning. Machine learning mostly studies ‘spatial’ dependencies between  $x_t$  and  $y_t$  (see Subsect. 4.2) while the theory of time series concentrates on the evolution of  $y_t$  with time.

Very simple methods of predicting time series are used, namely, predict the last element and exponentially weighted moving average. Still the results turn out to be comparable to a much more sophisticated kernel ridge regression technique. The experiments in this thesis may be seen as a follow-up to [21], where the same datasets were used.

Suppose that we want to apply a time series prediction method. The simplest way of doing this is to treat the outcomes  $\omega_1, \omega_2, \dots$  as a time series ignoring the signals  $x_t$  altogether. Obviously this can lead to a loss of potentially useful information.

Let us take a more refined approach and consider a partition of the domain  $X = \cup_{k=1}^K X_k$ , where  $X_i \cap X_j = \emptyset$  if  $i \neq j$ . This partition allows one to split the time series into  $k$  different series. The predictor then can work as follows. On step  $t$  it finds  $k$  such that  $x_t \in X_k$ , then picks the subsequence  $t_1 < t_2 < \dots < t_s < t$  such that  $x_{t_i} \in X_k$  and uses the time series  $\omega_{t_1}, \omega_{t_2}, \dots, \omega_{t_s}$  to make a prediction for step  $t$ . We will call this the *partitioning method*.

A disadvantage of this method is that it ignores all dependencies among  $\omega_t$  for different  $k$ . The moment  $t_s$  when the signal belonged to  $X_k$  for the last time may have occurred long ago compared to  $t$  and the outcomes may have evolved significantly since then.

This motivates the following algorithm. Consider a finite subset  $\{U_1, U_2, \dots, U_K\} \subseteq$

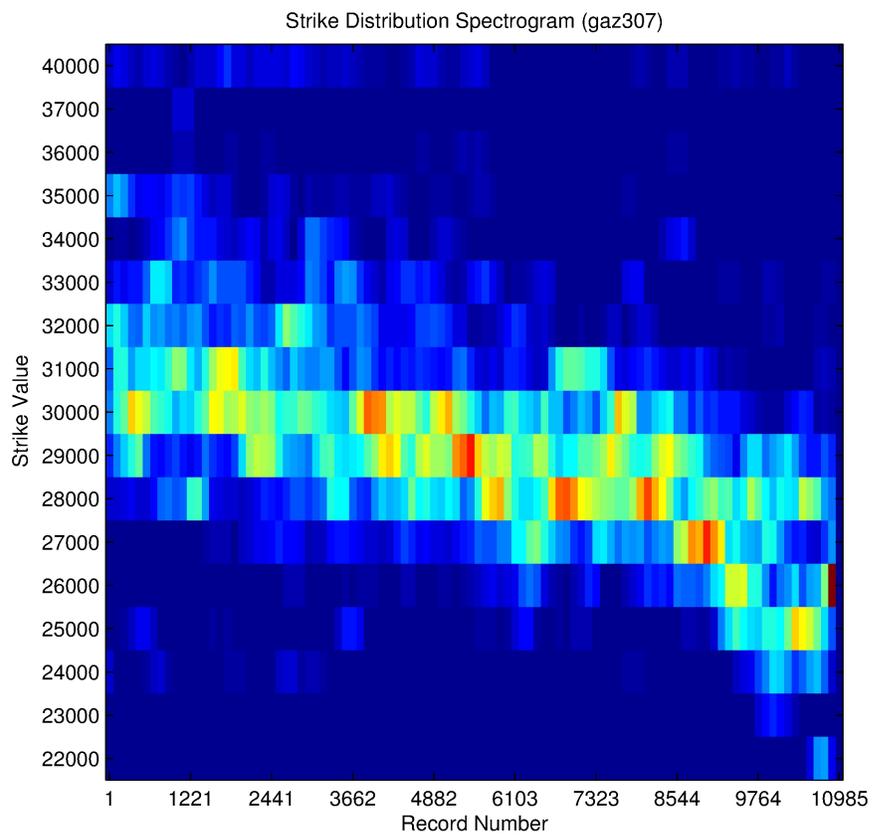


Figure 6.5: Distribution of Strikes (gaz307)

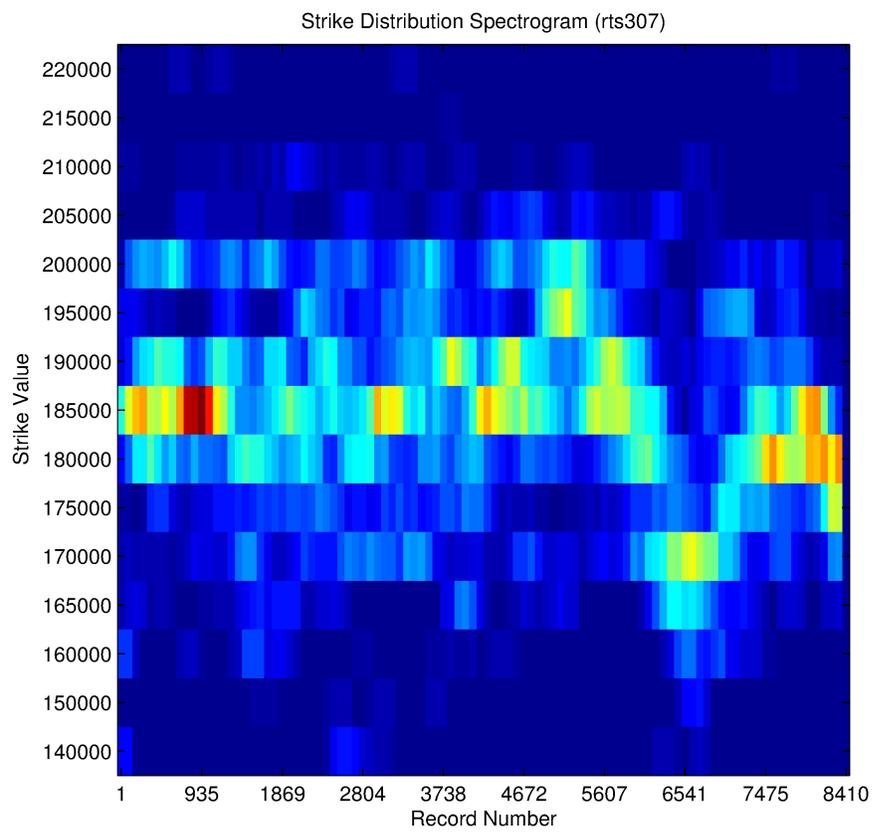


Figure 6.6: Distribution of Strikes (rts307))

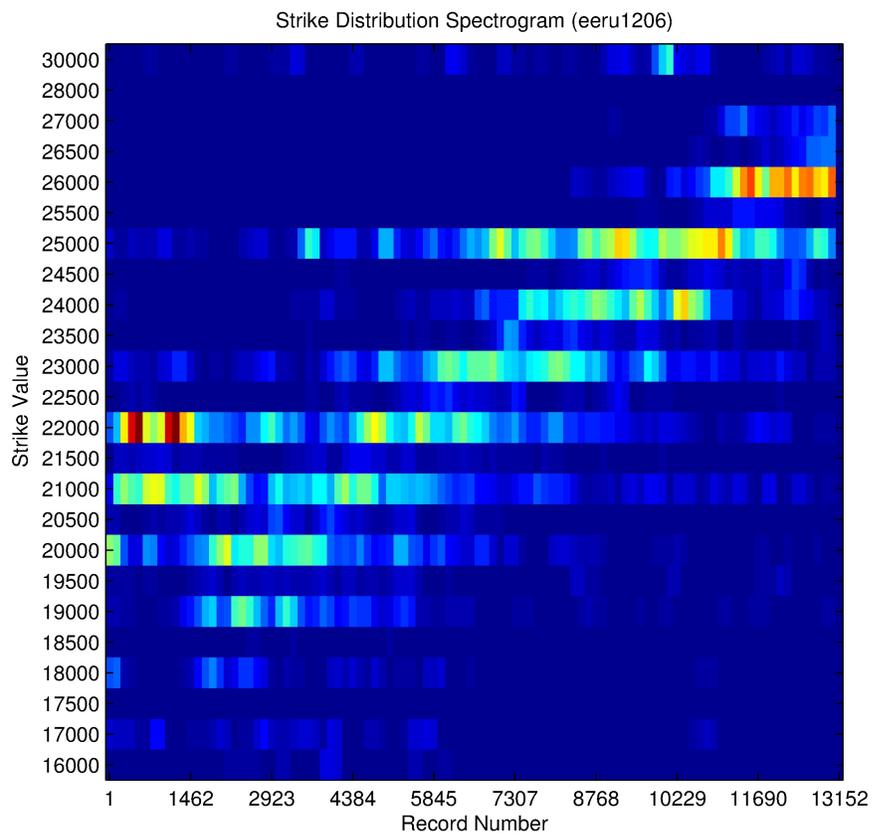


Figure 6.7: Distribution of Strikes (eeru1206)

$2^X$  such that  $\cup_{k=1}^K U_k = X$ . We will call sets  $U_k$  vicinities because it is natural to choose them in such a way that elements of  $U_i$  are in some respect akin to each other. Each vicinity  $U_i$  generates a specialist expert that predicts as follows. The expert is awake only on steps  $t$  where  $x_t \in U_i$ . It maintains the series  $\omega_{t_1}, \omega_{t_2}, \dots$  of outcomes for such steps and uses the series to make predictions for steps  $t$  where  $x_t \in U_i$ . For  $t$  such that  $x_t \notin U_i$  the expert makes no predictions. The experts are then merged using the aggregating algorithm for specialist experts. We will call this the *merging method*.

The AAS will automatically converge on relevant vicinities.

The computational complexity of the merging method depends on the underlying time series prediction algorithm. The extra complexity brought about by merging depends on the employed loss function; however, for most natural loss functions including the square loss merging takes time linear in the number of vicinities.

As an empirical study of an application of the theory of prediction with expert advice, this algorithm can be compared to [131], which also deals with the square loss function. Such applications are relatively few and may serve as testing grounds for method of prediction with expert advice in the future.

### 6.1.1 Side-Domain Time Series Merging Algorithm

In this section we describe an application of the algorithm of the thesis to the problem.

Let  $S$  consisting of  $s_1 < s_2 < \dots < s_L$  be the strikes for a dataset. A simple vicinity of diameter  $d$  is a set of  $d$  consecutive strikes; there are  $L - d + 1$  vicinities of diameter  $d$ .

A compound vicinity is a subset of  $S \times \{0, 1\}$ , where the 0/1 bit denotes whether the option in transaction is a put or call. A compound vicinity of diameter  $d$  is a product of a vicinity of diameter  $d$  by either 0 or 1; there are  $2(L - d + 1)$  compound vicinities of diameter  $d$ . Note that some of them may give rise to empty time series if, say, there were no transactions on put options with particular strikes. However every transaction belongs to at least one compound vicinity.

In the experiments below we took all simple and compound vicinities of diameters from 1 to  $d$  with  $d = 5$  (see Figure 6.4 for justification).

We are interested in the performance w.r.t. the squared loss so we apply the AAS for square loss. In order to apply AAS, predictions of time series methods were capped at 1 and thus for the purpose of merging we assumed that the prediction and outcome space were  $[0, 1]$ .<sup>2</sup> Equal initial weights were given to all experts.

Two very simple methods for predicting time series turned out to be very successful, predict the last element and exponentially weighted moving average (EWMA). In the former given a series  $y_1, y_2, \dots, y_T$ , the value  $\gamma_{T+1} = y_T$  is predicted. In the later the value  $\gamma_{T+1} = \lambda y_T + (1 - \lambda)\gamma_T$  is predicted, where  $\gamma_T$  is the prediction output on the previous step and  $\lambda$  is a parameter (taken to be 0.95). At the beginning where no previous element is available, we used the default value of 0.3.

For the purposes of prediction the values of volatility were capped at 1. The prediction algorithms worked on capped outcomes and thus produced predictions only from the interval  $[0, 1]$ .

The results of the prediction methods were evaluated against the loss of the proprietary RTSSE technique discussed in Subsect. 5.1.2 called the competitor loss for brevity. In the figures below (6.8-6.13) we plot the *adjusted loss* of algorithms, which is the cumulative loss minus the cumulative loss of the competitor. Note that for the purpose of calculating the loss, the outcomes were not capped at 1.

In line with the prequential approach of Dawid (see [38]) we do not assume a probabilistic model on the data and evaluate the quality of prediction using the cumulative loss on the actual datasets.

Figures 6.8–6.13 show the adjusted loss of predict the last element and EWMA in two modes, partitioning and merging. The solid lines represent the results of merging experts based on all simple and compound vicinities of diameters from 1 to 5. The dotted lines represent the naive partitioning mode with compound vicinities of diameter

<sup>2</sup>Tighter limits would allow one to choose a better learning rate  $\eta$ .

Table 6.3: Cumulative loss at the end of the dataset

Dataset	Competitor loss	Improvement achieved by			
		Predict last element		EWMA	
		Naive partitioning	Merging	Naive partitioning	Merging
eeru1206	185.05	13.91	25.25	15.13	<b>25.73</b>
gaz307	48.56	2.71	5.3	3.75	<b>6.19</b>
rts307	156.33	0.65	5.31	0.76	<b>5.56</b>

size 1. No merging happens there: a signal  $x_t$  uniquely determines a vicinity.

Table 6.3 shows the figures of cumulative adjusted loss at the end of the datasets.

### 6.1.2 Selecting the Range of Sizes

In our experiments above we took all compound vicinities of sizes from 1 to 5. Let us discuss the choice of the maximum size. Table 6.4 gives improvements over the competitor at the end of datasets for EWMA. Each line shows the result for the merging algorithm with simple and compound vicinities of diameters from 1 to the maximum size in the first column.

Number 5 is not necessarily the best maximum size, but the improvement brought about by larges sizes is very small. This comes at a cost of increased running time. For very large sizes the performance goes down, but very slowly.

This behaviour is easy to explain theoretically. The regret term in (4.8) is proportional to  $\ln(1/p_0(\theta))$ . Since we took a uniform initial distribution on experts, the regret is proportional to the logarithm of the number of experts.

The number of simple vicinities of diameter  $d$  is  $L - d + 1$ , where  $L$  is the number of strikes, so the total number of vicinities grows approximately linearly with  $d$ . The regret term thus grows approximately logarithmically. This growth is slow.

The conclusion is that loss-wise one should not be afraid to take large maximum diameter. In practice the main constrain restricting the growth of  $d$  is running time.

It is remarkable that vicinities of size *exactly*  $d$  perform poorly. For example, the

Table 6.4: Improvements of EWMA for different ranges of vicinities

Maximum size	eeru1206	gaz307	rts307
1	19.9	4.42	1.73
2	23.04	6.23	3.98
3	24.64	6.3	4.93
4	25.3	6.23	5.37
5	25.73	6.19	5.56
6	25.92	6.09	5.61
7	26.03	6.03	5.61
8	26.11	5.96	5.58
9	26.15	5.86	5.56
10	26.17	5.79	5.54
11	26.19	5.71	5.51
12	26.19	5.66	5.49
13	26.18	5.62	5.46
14	26.17	5.6	5.44
15	26.15	5.58	5.43

merging algorithm with EWMA running on simple and compound vicinities of diameter exactly 5 brings improvement 18.91 for eeru1206, 3.82 for gaz 307, and 2.69 for rts307. These results are not very good and worse than line 4 in Table 6.4; however adding vicinities of diameter 5 to vicinities of diameters 1 to 4 helps.

The conclusion is that in the mixture there should be vicinities of different sizes. AAS automatically picks up the right ones.

### 6.1.3 Selecting Types of Vicinities

In our experiments above we took both simple and compound vicinities. What happens if we drop compound vicinities?

Figure 6.14 shows the adjusted loss of EWMA in two modes, partitioning and merging, on eeru1206. The naive partitioning algorithm works on simple vicinities of diameter 1 and the merging algorithm on vicinities of diameters 1 to 5.

The performance is dramatically inferior to that on compound vicinities. Note

however that the merging algorithm still outperforms the naive.

It appears that the put/call bit makes a lot of difference to prediction. This indicates an interesting domain property. Theoretically for European options implied volatility should be the same for puts and calls with equal parameters; this follows from the put-call parity [35, 70, 80], which is a very general statement. The difference may be due to the fact that the options are actually American. Further investigation of this would be very interesting.

The argument in Section 16.1 of [71] shows that the implied volatility of a European call option is always the same as the implied volatility of a European put option when the two have the same strike price and maturity date.

## 6.2 Merging Spatial Dependencies

In this section, several forecasting algorithms will be introduced that use specialist expert techniques to merge between experts that model spatial dependencies in the corpus.

### 6.2.1 Merging Fixed Region Experts

Let us henceforth propose a new algorithm conceptualized in Figure 6.2.1 attempting to take advantage of the temporal structures observed and discussed in Section 5.3. In this section we will motivate and describe an algorithm that uses kernel ridge regression experts that have some time-dependent knowledge.

$R$  equally-sized ridge regression models with window size  $w$  are equally-spaced throughout the corpora, positioned from  $\max(c - w + 1, 1)$  to  $c$  for each  $c$  in

$$c_i = \left\lceil \left\{ \lambda^{\frac{N-w+1}{R}}, 2^{\frac{N-w+1}{R}}, \dots, \mathcal{R}^{\frac{N-w+1}{R}} \right\} \right\rceil$$

where  $N$  is the number of examples in the corpus. Note that each region expert starts outputting predictions only after its region has fully revealed itself; therefore it is

interesting to deal with sleeping experts algorithms [26] as introduced in Section 4.5.4 in addition to the fixed- and variable-share algorithms discussed in Section 4.5.8.

A  $N \times R$  prediction matrix is generated and fed into the specialist expert algorithm.

Depending on the window size and the number of region experts specified; it is possible for the regions to be overlapping.

The best set of parameters according the stochastic parameter search protocol introduced in Section 6.2 are presented in Table 6.6. Note that this means that the algorithm was run on the validation set `rts1206` hundreds of times and the set of parameters that produced the lowest median sum squared deviation adjusted loss was selected. Note that the so-called *adjusted loss* is the loss of the RTSSE competitor minus the method presented. The genetic search parameters are the same as those cited in Section 6.2.

The idea is that when experts from the past have something to say about the current regime, they will get more weight assigned to them by the specialist expert algorithm. The more contiguous the regions are, the less switching overhead should be present. Recall that the loss is a function of switching parameter  $\alpha$  and the number of experts  $\Theta$  (see Section 4.5.8).

We refer to the fall-back predictor  $f$  as an academic exercise. Predictions will only be issued for records  $w + 1, \dots, N$  so that we can evaluate the performance of our method independently of  $f$ . Furthermore; the corpora are very large in size and the predictive performance is more relevant near maturity when the nature of the data becomes increasingly chaotic. An example of a potential parsimonious fall-back predictor that required almost no training records would be simply predicting the last element or a time-series model such as ARIMA (see [97, 102, 132]).

Refer to the following subsection for discussion of the Figures produced by this algorithm.

See Figures 6.16, 6.17 and 6.18 for a plot of the cumulative sum adjusted squared deviation on the corpora. See Figures 6.19, 6.20 and 6.21 for an image map of log-expert

weights against time for the fixed region merging algorithm using variable share. The parameters used are in Table 6.5.

### 6.2.2 Merging Lagged Region Experts

We hereby propose a modification of the region experts algorithm described in Subsect. 6.2.1. One of the natural concerns that arises from the protocol described in the previous section is that the specialist experts weights get initialized with a uniform prior. It may be the case that the new expert coming on stream does not have sufficient weight for its predictions.

Rather than modifying the prior and transitional probabilities (see Subsect. 4.5.8) of specialist expert algorithms, a workaround is to have experts that correspond to regions that lag in time by varying time-horizons. This will also make the transition between regions more featured rather than an expert suddenly coming on stream.

As the Cholesky factor update allows us to compute sliding regression in quadratic time (see Section 4.4, [112]), this is a worthwhile mutation.

The lagged region experts will be trained on windows that are evenly spaced on regions

$$\{t - w, \dots, t - 1\} + L_i$$

where lags variable

$$L_i = \left\lfloor \left\{ (R-1)\frac{w}{R-1}, (R-2)\frac{w}{R-1}, \dots, \cancel{R}\frac{w}{R-1}, 1 \right\} \right\rfloor.$$

The genetic search will again find the optimal parameters for the number of experts  $R$ , the maximum lag time horizon, degree  $d$ , window size  $w$  and  $\alpha$  from the median cumulative *adjusted* squared deviation on validation set `rts1206`.

See Figures 6.22, 6.23 and 6.24 for a plot of the cumulative sum adjusted squared deviation on the corpora.

### 6.2.3 Merging Ridge Models with Variable Window Sizes

The forecasting algorithm henceforth referred to as the variable window size algorithm will merge  $R$  instances of sliding kernel ridge regression with  $R$  different window sizes ranging on a linear space from a specified minimum window size, to the maximum window size.

The genetic search will again find the optimal parameters for the number of experts  $R$ , the minimum window size, degree  $d$ , window size  $w$  and  $\alpha$  from the median cumulative *adjusted* squared deviation on validation set `rts1206`.

See Figures 6.25, 6.26 and 6.27 for a plot of the cumulative sum adjusted squared deviation on the corpora.

### 6.2.4 Stochastic Ridge Models

This section will describe three algorithms that in one way or another merge together instances of sliding ridge regression that have not be validated. Rather, their parameters have been randomly-generated on a reasonable (given the domain and the scaling of the data) and continuously distributed interval of pre-set values.

The intuition is that; as we have demonstrated from the visualization of the temporal structures in Section 5.2.4, the dependencies are changing through the corpus as a function of time. This means that different parts of the corpus require evolving parameters. By combining all possible parameters we are describing all possible configurations of dependencies. A key advantage of this strategy is that the resulting algorithms are validation-free and as a result require no training whatsoever for the basic parameters such as degree  $d$  and ridge coefficient  $a$ . The prediction with expert advice algorithm is relied upon to react to changing regimes.

For each corpus in the corpora, 500 sets of implied volatility predictions  $P_i$  were cached from these randomly generated kernel ridge regression models. The parameters of those models were randomly generated using the following criteria. The degree  $d$

was generated from the discrete set

$$\{1, 2, \dots, 10\},$$

window sizes  $w_i$  in

$$\{50, 80, 110, \dots, 230\}$$

and ridge coefficients  $a_i$  in

$$\{0.1, 0.2, 0.3, \dots, 10\}.$$

#### 6.2.4.1 Average Random Ridge Regression Algorithm

This algorithm will select all 500 of the pre-computed prediction sets from  $P_i$  and outputs the mean average of all the pre-computed predictions for each record. Because there are so many sets of parameters being selected, the result of the algorithm is almost exactly the same every time with negligible variation ( $\leq 1e^{-3}$ ).

#### 6.2.4.2 Average Merged Random Ridge Regression

This algorithm will  $S$  times, select  $R$  pre-computed prediction sets from  $P_i$  and merge them together using the specialist expert techniques. The  $S$  sets of predictions will then be averaged using the mean-average.  $S$  will be referred to as the *stack count*. The genetic algorithm protocol still applies here and will search for optimal parameters  $S$  and  $R$  and  $\alpha$ . This algorithm is also non-deterministic. With increasing  $S$  and  $R$ ; the variability of the algorithm will increasingly reduce.

#### 6.2.4.3 Meta-merging

This algorithm is the same as the one previously described with the exception that the predictions of the specialist expert techniques will themselves be merged together again using the specialist expert algorithms.

See Figure 6.2.4.3 for an illustration of the amount of variability for 100 calls to

the algorithm on corpus `gaz307` given the optimal parameters found with the genetic search. The standard deviation for the median metric was 0.008 which is an acceptable level of deviation given the scale of change on the other results in Table 6.6.

Refer to Figures 6.29, 6.29 and 6.29 for plots of the adjusted squared deviation for the average merged random ridge regression mentioned in Section 6.2.4.2 and the meta-merging algorithm described in Section 6.2.4.3 compared against the sliding window kernel ridge regression algorithm.

### 6.3 Final Parameters & Results

See Table 6.6 for the final table of results comparing the forecasting performance of all the algorithms introduced in this chapter on the RTSSE corpora.

See Table 6.5 for the parameters of the algorithms introduced in this chapter on the RTSSE corpora as found by the genetic search described in Section 6.2.

In the main table of results, the median measurement is a strong indication of how the algorithm performs for most of the time. There are several outliers and some chaotic activity near maturity (as discussed in Section 5.2) which means the `Net` metric (the sum adjusted loss) is a strong indicator of how resistant the method is to outliers and how regularised it is also.

These results will be discussed in detail in Chapter 7.

Table 6.5: The parameters used on the algorithms as found by the genetic searches described in Section 6.2. *Net* means the sum adjusted deviation of all predictions. Median is of the cumulative sum adjusted deviation.

	Degree $d$	Window $w$	Ridge $a$	$\alpha$	Max Time Horizon	Min Window	# Experts	Stack Count
Sliding Window RR	7	355	7.70					
Fixed Region Fixed-Share	5	294	2.99	0.38			134	
FixedRegion Variable-Share	5	286	0.48	0.96			83	
Fixed Region Sleeping	5	293	1.83				146	
Lagged Region Fixed-Share	6	205	5.50	0.21	240		14	
Lagged Region Variable-Share	5	111	1.46	0.99	536		26	
Lagged Region Sleeping	5	244	1.71		209		52	
Variable Window Fixed-Share	5	242	1.25	0.25		40	120	
Variable Window Variable-Share	7	246	0.35	0.87		40	124	
Variable Window Sleeping	5	250	1.45			94	79	
Average Random-RR								
Average Merged RRR FS				0.65			26	7
Average Merged RRR VS				0.86			40	17
Average Merged RRR SE				0.78			249	7
Meta-merged RR VS				0.72			100	16
Meta-merged RR VS				0.99			119	8
Meta-merged RR SE				0.38			62	5

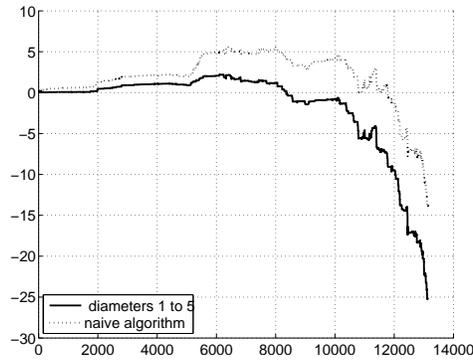


Figure 6.8: Predict last element on eeru1206

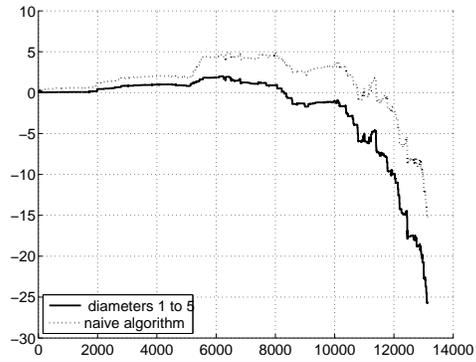


Figure 6.9: EWMA on eeru1206

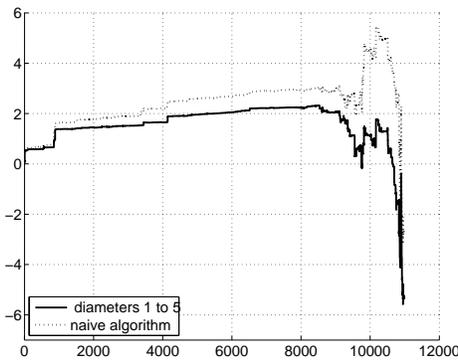


Figure 6.10: Predict last element on gaz307

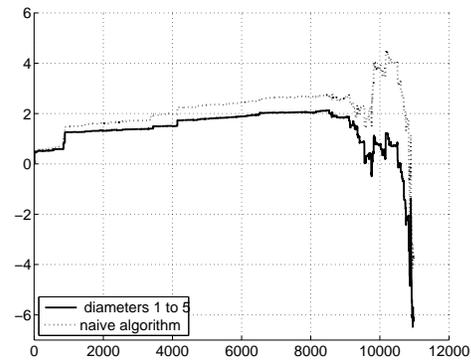


Figure 6.11: EWMA on gaz307

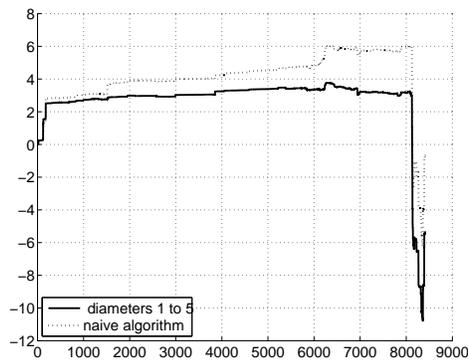


Figure 6.12: Predict last element on rts307

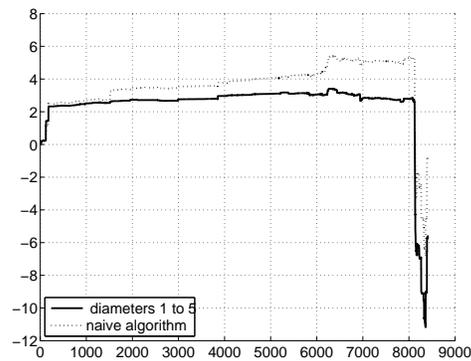


Figure 6.13: EWMA on rts307

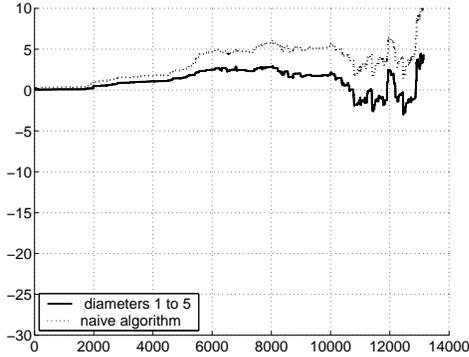


Figure 6.14: EWMA on euru1206 with simple vicinities

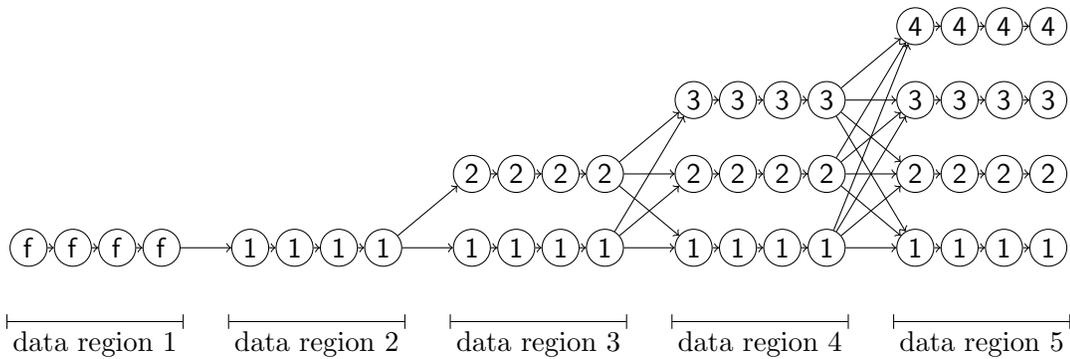


Figure 6.15: Expert Hidden Markov Model Rendering of Fixed Share with Region Experts. A fallback predictor  $f$  would be required to predict region 1. Each expert  $i$  is a ridge regression predictor trained on data region  $i$ . The transition probabilities for fixed- and variable-share are specified in Subsect. 4.5.8.

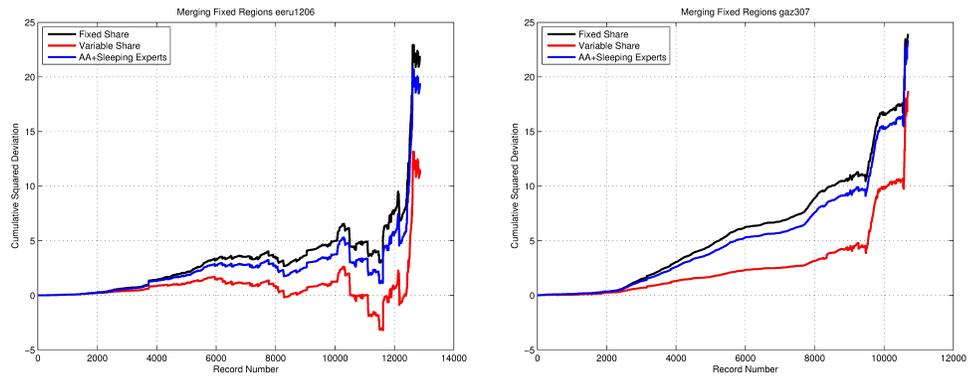


Figure 6.16: Merging fixed regions on eeru1206  
Figure 6.17: Merging fixed regions on gaz307

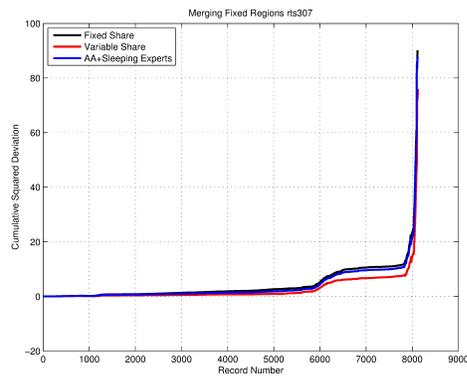


Figure 6.18: Merging fixed regions on rts307

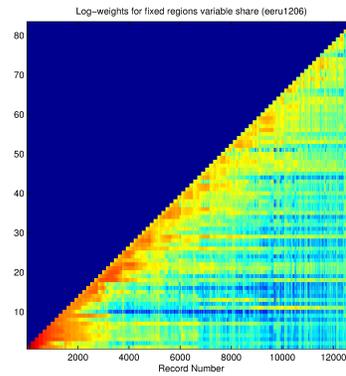


Figure 6.19: Merging fixed regions log-weights on `eeru1206`. Note that the y-axis is the region expert predictor. Oranges are low expert weight, blues are high weight.

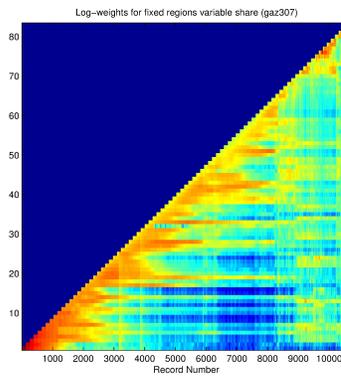


Figure 6.20: Merging fixed regions log-weights on `gaz307`.

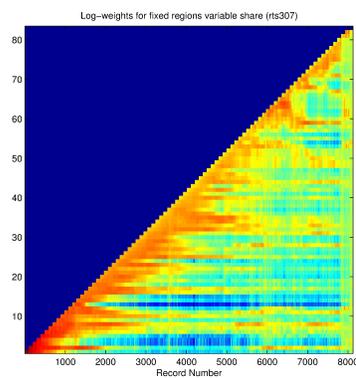


Figure 6.21: Merging fixed regions log-weights on `rts307`.

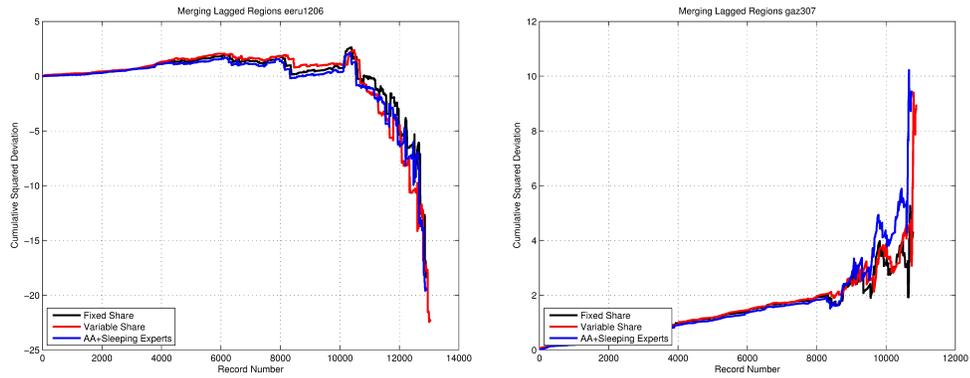


Figure 6.22: Merging lagged regions on eeru1206  
 Figure 6.23: Merging lagged regions on gaz307

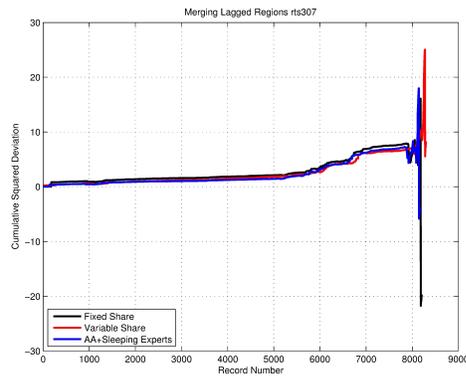


Figure 6.24: Merging lagged fixed regions on rts307

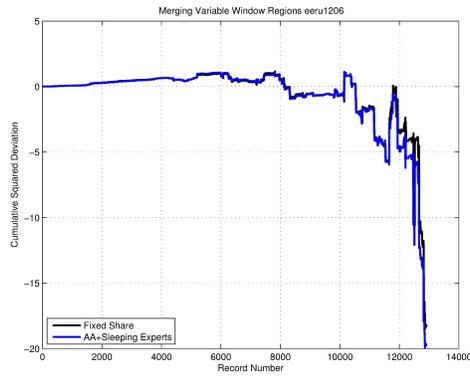


Figure 6.25: Merging variable window size experts on eeru1206

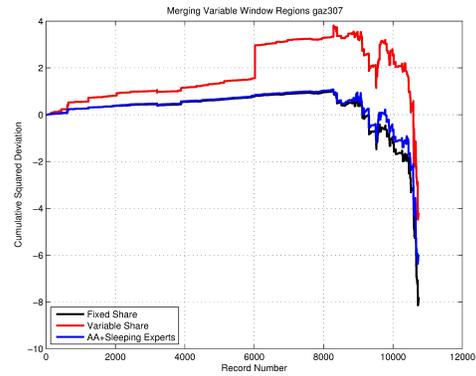


Figure 6.26: Merging variable window size experts on gaz307

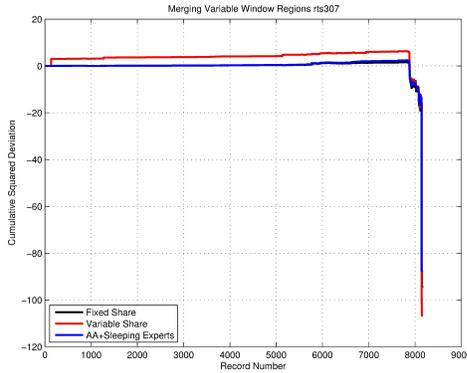
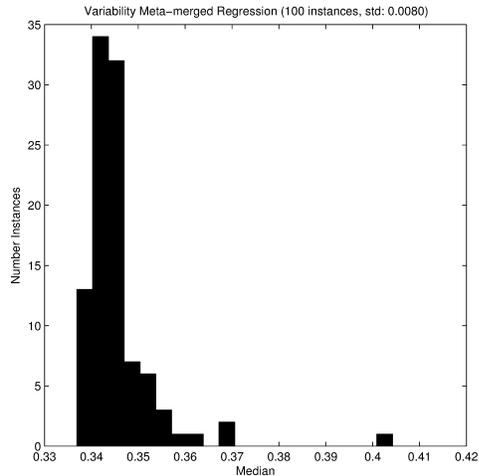


Figure 6.27: Merging variable window size experts on rts307

Figure 6.28: The amount of variability on the Meta-merging regression algorithm with the optimal parameters  $R = 119$ ,  $S = 8$ ,  $\alpha = 0.99$ .



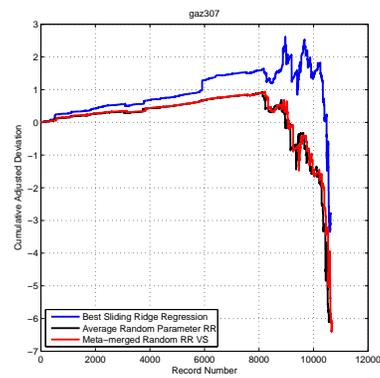
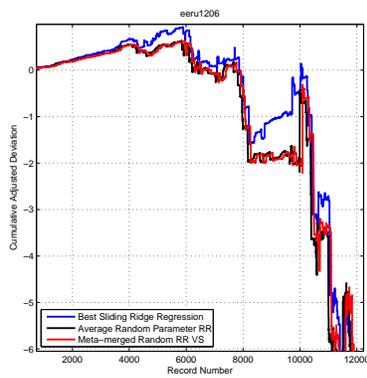


Figure 6.29: Merging random ridge regression models and sliding kernel ridge regression competitor on eeru1206

Figure 6.30: Merging random ridge regression models and sliding kernel ridge regression competitor on gaz307

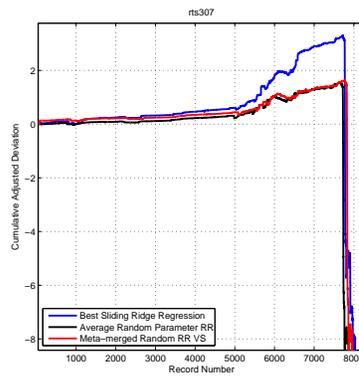


Figure 6.31: Merging random ridge regression models and sliding kernel ridge regression competitor on rts307

Table 6.6: The final results table of the forecasting algorithms introduced in Chapter 6. **Net** means the sum adjusted loss and **Median** is the median cumulative sum adjusted loss (see Section 6.1.1). Results of interest are in bold and are discussed in the conclusions (see Chapter 7).

	eeru1206		gaz307		rts307	
	Net	Median	Net	Median	Net	Median
Sliding Window RR	-32.04	<b>0.09</b>	-2.78	0.81	-36.81	0.44
Fixed Region Fixed-Share	21.93	3.19	23.94	5.21	90.15	1.91
Fixed Region Variable-Share	11.49	0.66	18.69	1.93	75.98	<b>0.75</b>
Fixed Region Sleeping	19.43	2.24	23.33	4.39	87.73	1.41
Lagged Region Fixed-Share	-18.17	0.61	4.30	1.29	-19.69	1.85
Lagged Region Variable-Share	-22.34	0.98	8.96	1.33	8.26	1.53
Lagged Region Sleeping	-19.45	0.44	9.44	<b>1.20</b>	-3.77	1.25
Variable Window Fixed-Share	-18.24	0.29	-7.83	<b>0.44</b>	-94.32	<b>0.21</b>
Variable Window Variable-Share	78.41	7.46	-4.22	1.36	-106.61	3.98
Variable Window Sleeping	-19.69	0.26	-6.02	0.47	-87.54	0.26
Merging Time Series PLE+Merge	-25.25	0.21	-5.30	1.27	-5.31	0.79
Merging Time Series PLE+Partition	-13.91	2.56	-2.71	2.00	-0.65	0.96
Merging Time Series EW+Merge	-25.73	<b>0.14</b>	-6.19	1.15	-5.56	0.68
Mixing Time Series EW+Partition	-15.13	2.29	-3.75	1.88	-0.76	0.87
Average Random-RR	-26.76	<b>0.06</b>	-4.86	<b>0.37</b>	-65.75	<b>0.24</b>
Average Merged RRR FS	-7.46	16.25	7.23	13.51	-63.97	9.45
Average Merged RRR VS	-24.18	<b>0.02*</b>	-5.87	<b>0.33</b>	-72.19	<b>0.20*</b>
Average Merged RRR SE	1.34	25.24	5.85	12.33	-61.81	10.09
Meta-merged RR VS	-15.52	8.51	-4.14	2.17	-67.38	3.87
Meta-merged RR FS	-24.19	<b>0.02*</b>	-6.29	<b>0.32*</b>	-70.93	0.32
Meta-merged RR SE	-18.86	5.29	-3.37	3.05	-70.78	0.33

## Chapter 7

# Conclusions

We have explored temporal structures based on self-similarity in two different contexts. First of all the development of an efficient dynamic programming algorithm for discovering the temporal structure of music shows and obtaining high quality results which were compared to humans in respect of the same task and the most popular algorithm in the literature.

Secondly; we have shown that prediction with expert advice techniques can be used to exploit temporal structure of the data for prediction. In this chapter we will discuss the results of the thesis and future research directions.

In this chapter we will explore our results in the context of this rubric.

### 7.1 Discovering Temporal Structures In Music

We believe the music segmentation algorithm would be useful for segmenting DJ-mixed audio streams in batch mode. The algorithm segments a 2 hour long show with all the cost matrices enabled in less than 2 seconds on our machine with a tile size of 3, implemented in MATLAB (including loading the `wave` file from the hard drive). Note that this does not include the conversion time to `wave` from `mp3`. MATLAB includes features to perform executions on multiple cores and the GPU which we have used for

running the genetic algorithm parameter search.

It would be excellent if *SoundCloud*<sup>1</sup> or *MixCloud*<sup>2</sup> for example started to do something similar. *SoundCloud* and *MixCloud* are on-line music services with many electronic dance music radio shows with the track listing in text. This method would allow them to reliably segment the shows, and they could display an interactive segmentation in the music player with the track names annotated. Their music players could also be adapted to show temporal structures from the self-similarity matrix to convey some information about the technical style of mixing from the DJ and the similarity between tracks.

It is important to note that the cost matrices and high level algorithm are not encoded with any domain specific knowledge pertaining to dance music. They are looking for abstract and obvious patterns of self-similarity, which would be present in nature. We would expect this algorithm to perform well on any segmentation task of similar description (for example, video) with a different set of features to build the similarity matrix from.

The new cost matrices in combination improve robustness significantly over a summation cost matrix alone. Rather than improving the time accuracy; they eliminate many circumstances in which tracks get placed in erroneous order. They are also partly immunised against dissimilar regions within tracks which was a weakness in [109]. One problem that we are aware of is the rare instances when there are head or tail segments to a track that seem independent from the rest of the track. When these are small they usually get absorbed without any problems but they can cause misplacements. Preprocessing  $S_{ij}$  using heuristics to remove these *pariah* segments is potential solution.

We are able to operate in the scenario when the number of tracks is not known a priori and perform comparably with our *enhanced* version of Foote's [49] method for both segmentation and estimation of how many tracks exist in a recording. When

---

<sup>1</sup><http://www.soundcloud.com>

<sup>2</sup><http://www.mixcloud.com>

Foote's method was implemented by its literal description it performed quite poorly. On the `lindmik` dataset, we out-perform the enhanced Foote method when estimating the number of tracks. Our method comes into its own however in the scenario when the number of tracks is known a priori. We significantly out perform Foote's method for this and there is no constructive way to adapt Foote's method to find a fixed number of tracks. In this task it is essential to avoid getting the order of tracks wrong, so any potential advantage should be capitalised on.

## 7.2 Forecasting on RTSSE Corpora

See Table 6.6 for the main set of results and Table 6.5 for the parameters found from the genetic searches.

We have introduced algorithms that significantly beat the RTSSE on the sum squared deviation (the loss at the end of the corpora). And out-perform kernel ridge regression on average (according to the median metric). And in the case of the merged stochastic models (Section 6.2.4) and the merged time-series models (Section 6.1) the algorithms are essentially parameterless and do not require any validation.

Please see the Figures A.1-A.16 in Chapter A for the visual output from MATLAB. The bottom left panel shows the variation in the objective function for the last random mutation in generation 5. It is a good indication of how regular, convex or chaotic each model problem is. For example; the random model meta-merging in Figure A.15 is highly chaotic and standard sliding ridge regression is likely convex (see Figure A.6).

### 7.2.1 Merging Temporal Models

We have proposed a computationally simple algorithm for on-line prediction based on the use of specialist expert techniques that does not make any assumptions about the stochastic mechanism that generated the data and is validation-free. The empirical results show that the algorithm is comparable with kernel ridge regression and the

RTSSE technique on the entire corpora.

The average-merged random ridge regression method (variable share) performs the best of all.

See Figures 6.8-6.13 for the comparative performance of the merging time series algorithm, and Figures 6.2,6.4 and 6.3 for kernel ridge regression as a comparator.

The figures and the tables cited lead to the following conclusions.

1. Simple time series methods applied strike-wise perform comparably to the RTSSE proprietary technique. At the end of the datasets (i.e., when the options approach maturity) they outperform the proprietary technique.

The figures show that the competitor outperforms our time-series merging methods at the beginning. A plausible explanation is that the competitor incorporates some prior knowledge about the behaviour of volatility, while our methods need to learn from scratch. It is tempting to attribute our good performance at the end to learning the data well. However there is a more realistic explanation. The period when we outperform the competitor is rather short; it is even shorter in terms of ‘physical’ time (transactions happen much more frequently near the end). Presumably the competitor was optimised to perform well *most of the time*.

2. EWMA with  $\lambda = 0.95$  performs slightly but consistently better than predict the last element.

3. Merging method with compound vicinities of diameters from 1 to 5 performs consistently better than naive partitioning. It is better most of the time with the gap widening with time.

It should be noted here that the methods perform very fast and are validation-free. Neither the underlying methods nor the sleeping algorithm take much time. On a typical pc the running time goes from seconds to single minutes as  $d$  is increased from 1 to 15.

For the first two datasets (`eeru1206` and `gaz307`) the performance of regression is comparable to our method; our methods outperforms ridge regression on `gaz307` in

terms of the cumulative loss at the end of the dataset. For the `rts307` corpus ridge regression greatly outperforms our method, but it is due to a very short interval at the end.

It is interesting that ridge regression uses much more information than our method. The merging algorithm in our set-up makes predictions only on the basis of strike and put/call bit. This hints at an interesting property of the domain. This property requires further investigation.

### 7.2.2 Merging Spacial Models

Figures 6.21, 6.20 and 6.19 clearly show that the temporal regime structures as described qualitatively in Sections 5.2.4, 5.3 and 5.4 are taking an effect on the expert weights in the fixed regions algorithm with variable share. The temporal structures look extremely similar to those discovered in the music corpora in Chapter 3 (see Figure 3.11). The median score was comparable to sliding kernel ridge regression but did not beat it. The performance on `rts307` was particularly good and made sense given that in Figure 5.11 we demonstrated that it had a high degree of forward-predictability from region A. The other two corpora did not have the same degree of forward-predictability. Figure 5.11 also demonstrates that weights can be recovered in the future even after they have been given a low value.

The two likely reasons that the performance is generally lower than sliding ridge regression is that there is an overhead associated with switching (linked to  $\alpha$  and the number of experts  $\Theta$ , see Section 4.5). It is also possible that while old information is highly useful in certain circumstances, recency is just as important.

A possible theory is that the regions algorithm places too much emphasis on spatial dependencies. A region can only start to make predictions when it is fully revealed; and this means that very recent information is not available most of the time. It does however have some interesting properties; it only requires a small number of ridge regression models rather than needing constant re-training.

The lagged regions algorithm performs similarly to the fixed-regions algorithm. The expectation for this algorithm was that the performance would be higher because there was no switching cost to move to an old region unless it is actually producing good predictions. In the average case when the most recent sliding window is predicting the best, it can receive all of the weight.

As the fixed-share scheme is the best on the lagged regions algorithm, it indicates the protocol is less switch-dependent. What happens is that the region that is most recent just keeps almost all of the weight.

Because the fixed region algorithm is highly switch-dependent it is no surprise that the variable-share algorithm beats the other two. When a new region reveals itself, it will presumably be making strong predictions but under the other two schemes it would not be able to get weight fast enough.

See Figures 7.1-7.3 for image maps of the expert weights against time on the corpora for the merging variable windows algorithm (see Section 6.2.3). These images tell quite a story of when long range information, and when immediate information is important. On `rts307`, any sized training window up to a fixed size is desirable until record 6500 (near maturity), then smaller windows get selected. On `eeru1206`, long range information is preferred and there is a region around 8000 records, and again near maturity when very large training windows are preferred. On `gaz307`, smaller training windows are rejected near maturity. This seems odd given that `gaz307` is unpredictable near maturity (see Figure 5.10).

Refer to Figures 6.29, 6.29 and 6.29 for plots of the adjusted squared deviation for the average merged random ridge regression mentioned in Section 6.2.4.2 and the meta-merging algorithm described in Section 6.2.4.3 compared against the sliding window kernel ridge regression algorithm.

Switching algorithms only provide a negligible improvement over averaging together the predictions of 500 randomly generated models. Increasing the number of randomly generated models did not provide further improvement, neither did it have a deleterious

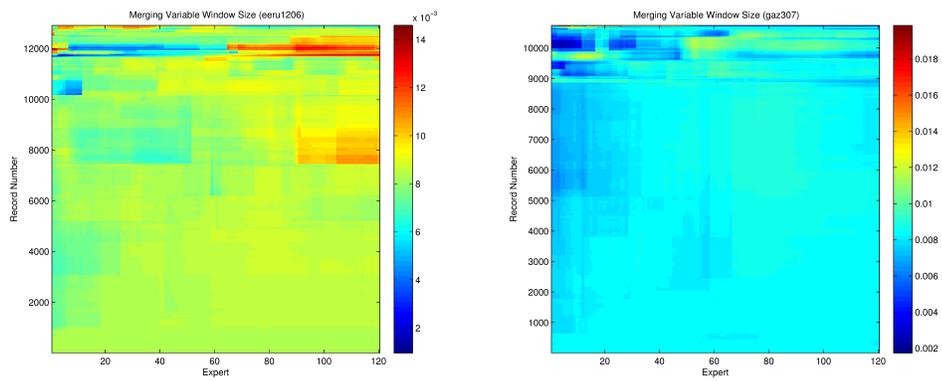


Figure 7.1: Variable windows expert weights (fixed-share) on euru1206      Figure 7.2: Variable windows expert weights (fixed-share) on gaz307

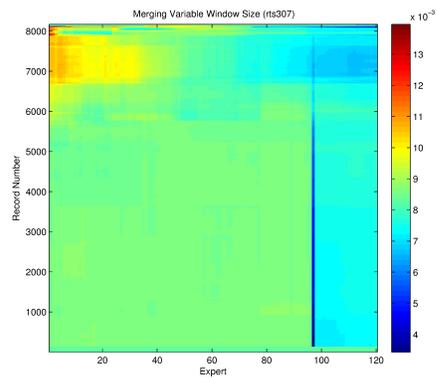


Figure 7.3: Variable windows expert weights (fixed-share) on rts307

effect on the performance.

The real question is why this works at all. Apparently there is little or no cost to pay for averaging over all the models, even if they do not perform well. Although a caveat is that only models with reasonable (but wide-ranging) parameters were selected (see Section 6.2.4).

LeBlanc reported in [83] that a combination of estimators (not predictions), used with some regularization, can be a useful tool both for improving prediction performance and for learning about the structure of the problem. LeBlanc et al derived and studied a number of procedures and found that cross-validation (model-mix) and the bootstrap, used with a nonnegativity constraint worked best.

Model combining is also explored by Yuan et al in [134], but there is no apparent work done with prediction mixing in this trivial fashion.

The algorithms are not deterministic, although the variance of the result becomes increasingly negligible when more models are mixed over (as reported in Section 6.2.4.3).

The methods of prediction with expert advice have shown their power in selecting good experts. The experiments demonstrate that they are capable of selecting good models out of a random selection of experts.

We have shown that on small random samples of experts the methods of prediction with expert advice are superior to simple averaging. For larger samples this effect fades.

### 7.3 Further Work

We are going to create an on-line version of the music segmentation algorithm in which will operate on a sliding lagged window of audio where the number of tracks could be estimated and the algorithm executed on the window. We would also like to implement a regularised version of Radu's time dependent agglomerative (hierarchical) clustering algorithm [37] to see if it is suitable for this task.

One of the interesting properties of the segmentation algorithm is that it does not

directly consider inter-segment dissimilarity. The costs are computed only from intra-segment similarity. Therefore; there is only an implied notion of dissimilarity between segments. In Section 5.4 we modified the music segmentation algorithm to consider transitional *switching* and *sticking* costs through a state graph where the number of switches was fixed a priori. This allowed us to use almost the same algorithm to find the least cost path through the similarity matrix given a fixed number of switches. This would be a somewhat similar direction to [58, 59] apart from the likelihood that fixing the number of segments a priori if they were known would likely improve the accuracy of the annotations as we have reported for this configuration.

The fixed region algorithm could possibly be improved by devising a probability scheme where it assumed that new experts know more and that when old experts know something again there is likely to be some vicinity of expertise i.e. the neighbours of the expert will also know something now or soon.

A potential research direction would be to vary the switching rate dynamically in respect of the observed recent variance of the volatility.

We would also like to construct an artificial dataset that demonstrates that the regions and lagged algorithm could beat the sliding ridge regression competitor when obvious, recurrent contiguous regions are present. We would also like to try a scheme dealing with a combination of region experts as in the fixed regions algorithm but also with sliding window expert. This would potentially give the benefit of recent information without any switch cost, but also old information if it were needed.

A similar version of the merging time series algorithm but using a more less parsimonious predictor (for example an ARIMA model [73, 102, 132]).

Comparison against a naive sliding ridge regression competitor that re-validated periodically. It is a disadvantage for ridge regression to be validated on another corpus, and also validated across the whole of the corpus when the dependencies clearly change as a function of time. We believe such an experiment would produce results similar to the stochastic parameter merging algorithms (see Section 6.2.4).

Using a different switching scheme from the literature, perhaps mixing past-posteriors by Bousquet et al [17] or a self-referential part of the past-posteriors as in [2].

# Bibliography

- [1] E. Aarts, J. Korst, and W. Michiels. Simulated annealing. In *Search methodologies*, pages 187–210. Springer, 2005.
- [2] D. Adamskiy, M. K. Warmuth, and W. M. Koolen. Putting bayes to sleep. In *Advances in Neural Information Processing Systems*, pages 135–143, 2012.
- [3] A. Aizerman, E. M. Braverman, and L. Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, 25:821–837, 1964.
- [4] S. An, W. Liu, and S. Venkatesh. Face recognition using kernel ridge regression. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–7. IEEE, 2007.
- [5] U. Appel and A. V. Brandt. Adaptive sequential segmentation of piecewise stationary time series. *Information sciences*, 29(1):27–56, 1983.
- [6] K. S. Azoury and M. K. Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning*, 43:211–246, 2001.
- [7] M. A. Babyak. What you see may not be what you get: a brief, nontechnical introduction to overfitting in regression-type models. *Psychosomatic medicine*, 66(3):411–421, 2004.

- [8] A. Baines. *Woodwind instruments and their history*. Courier Corporation, 1967.
- [9] C. Barras, X. Zhu, S. Meignier, and J.-L. Gauvain. Improving speaker diarization. In *RT-04F workshop*, 2004.
- [10] M. Basu. Gaussian-based edge-detection methods-a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 32(3):252–260, 2002.
- [11] E. Batlle, J. Masip, and E. Guaus. Automatic song identification in noisy broadcast audio.
- [12] J. Beament. *How we hear music: The relationship between music and the hearing mechanism*. Boydell Press, 2003.
- [13] J. M. Bennett. Triangular factors of modified matrices. *Numerische Mathematik*, 7:217–221, 1965.
- [14] C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.
- [15] F. Black and M. Scholes. The pricing of options and corporate liabilities. *The journal of political economy*, pages 637–654, 1973.
- [16] E. d. Boer and J. Bouwmeester. Critical bands and sensorineural hearing loss. *International Journal of Audiology*, 13(3):236–259, 1974.
- [17] O. Bousquet and M. K. Warmuth. Tracking a small set of experts by mixing past posteriors. *The Journal of Machine Learning Research*, 3:363–396, 2003.
- [18] S. I. Boyarchenko and S. Levendorskii. *Non-Gaussian Merton-Black-Scholes Theory*, volume 9. World Scientific Singapore, 2002.
- [19] S. Busuttil and Y. Kalnishkan. Online regression competitive with changing predictors. In *Algorithmic Learning Theory, 18th International Conference, Proceedings*, pages 181–195, 2007.

- [20] S. Busuttill and Y. Kalnishkan. Online regression competitive with changing predictors. In *Algorithmic Learning Theory*, pages 181–195. Springer, 2007.
- [21] S. Busuttill and Y. Kalnishkan. Weighted kernel regression for predicting changing dependencies. In *Machine Learning: ECML 2007*, pages 535–542. Springer, 2007.
- [22] G. C. Cawley, N. L. Talbot, and O. Chapelle. Estimating predictive variances with kernel ridge regression. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Textual Entailment*, pages 56–77. Springer, 2006.
- [23] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.
- [24] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.
- [25] S. Chen and P. Gopalakrishnan. Speaker, environment and channel change detection and clustering via the bayesian information criterion. In *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, page 8. Virginia, USA, 1998.
- [26] A. Chernov, Y. Kalnishkan, F. Zhdanov, and V. Vovk. Supermartingales in prediction with expert advice. *Theoretical Computer Science*, 411(29-30):2647–2669, 2010.
- [27] A. Chernov and V. Vovk. Prediction with expert evaluators’ advice. In *Algorithmic Learning Theory, ALT 2009, Proceedings*, volume 5809 of *LNCS*, pages 8–22. Springer, 2009.
- [28] A. Chernov and V. Vovk. Prediction with expert evaluators’ advice. In *Algorithmic Learning Theory*, pages 8–22. Springer, 2009.

- [29] A. Chernov and F. Zhdanov. Prediction with expert advice under discounted loss. In *Proceedings of ALT2010*, volume 6331 of *LNAI*, pages 255–269, 2010.
- [30] A. V. Chernov and F. Zhdanov. Prediction with expert advice under discounted loss. In *Proceedings of ALT 2010*, volume LNAI 6331, pages 255–269. Springer, 2010.
- [31] B. Chiu, E. Keogh, and S. Lonardi. Probabilistic discovery of time series motifs. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 493–498. ACM, 2003.
- [32] N. Chriss. *Black-Scholes and beyond: option pricing models*. Irwin, 1997.
- [33] M. L. Cooper and J. Foote. Automatic music summarization via similarity analysis. In *ISMIR*, 2002.
- [34] J. C. Cox, S. A. Ross, and M. Rubinstein. Option pricing: A simplified approach. *Journal of financial Economics*, 7(3):229–263, 1979.
- [35] M. Cremers and D. Weinbaum. Deviations from put-call parity and stock return predictability. 2010.
- [36] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.
- [37] R. Curticapean. Clustering-based audio segmentation with applications to music structure analysis.
- [38] A. P. Dawid. Present position and potential developments: Some personal views: Statistical theory: The prequential approach. *Journal of the Royal Statistical Society. Series A (General)*, pages 278–292, 1984.
- [39] S. de Rooij. Tracking the best predicting model. In *Recent Breakthroughs in Minimum Description Length Learning Workshop*, page 9.

- [40] F. X. Diebold, J.-H. Lee, and G. C. Weinbach. Regime switching with time-varying transition probabilities. 1994.
- [41] T. Dietterich. Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, 27(3):326–327, 1995.
- [42] J.-P. Eckmann, S. O. Kamphorst, and D. Ruelle. Recurrence plots of dynamical systems. *EPL (Europhysics Letters)*, 4(9):973, 1987.
- [43] D. El Badawy, P. Marmaroli, and H. Lissek. Audio novelty-based segmentation of music concerts.
- [44] J. Foote. A similarity measure for automatic audio classification. In *Proc. AAAI 1997 Spring Symposium on Intelligent Integration and Use of Text, Image, Video, and Audio Corpora*, 1997.
- [45] J. Foote. Visualizing music and audio using self-similarity. In *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, pages 77–80. ACM, 1999.
- [46] J. Foote. Automatic audio segmentation using a measure of audio novelty. In *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, volume 1, pages 452–455. IEEE, 2000.
- [47] J. Foote and M. Cooper. Visualizing musical structure and rhythm via self-similarity. In *Proceedings of the 2001 International Computer Music Conference*, pages 419–422, 2001.
- [48] J. Foote and S. Uchihashi. The beat spectrum: A new approach to rhythm analysis. In *ICME*, 2001.
- [49] J. T. Foote and M. L. Cooper. Media segmentation using self-similarity decomposition. In *Electronic Imaging 2003*, pages 167–175. International Society for Optics and Photonics, 2003.

- [50] Y. Freund, R. E. Schapire, Y. Singer, and M. K. Warmuth. Using and combining predictors that specialize. In *Proceedings of STOC'97*, pages 334–343. ACM, 1997.
- [51] M. Frigo and S. G. Johnson. The fftw web page, 2004.
- [52] J.-L. Gauvain, L. Lamel, and G. Adda. Partitioning and transcription of broadcast news data. In *ICSLP*, volume 98, pages 1335–1338, 1998.
- [53] X. Ge and P. Smyth. Segmental semi-markov models for endpoint detection in plasma etching. *IEEE Transactions on Semiconductor Engineering*, 2001.
- [54] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders. Methods for modifying matrix factorizations. *Mathematics of computation*, 28(126):505–535, 1974.
- [55] D. E. Goldberg. E. 1989. genetic algorithms in search, optimization, and machine learning. *Reading: Addison-Wesley*, 1990.
- [56] S. M. Goldfeld and R. E. Quandt. Nonlinear methods in econometrics. 1972.
- [57] S. M. Goldfeld and R. E. Quandt. A markov model for switching regressions. *Journal of econometrics*, 1(1):3–15, 1973.
- [58] M. M. Goodwin and J. Laroche. Audio segmentation by feature-space clustering using linear discriminant analysis and dynamic programming. In *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on.*, pages 131–134. IEEE, 2003.
- [59] M. M. Goodwin and J. Laroche. A dynamic programming approach to audio segmentation and speech/music discrimination. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, volume 4, pages iv–309. IEEE, 2004.

- [60] J. D. Hamilton. A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica: Journal of the Econometric Society*, pages 357–384, 1989.
- [61] J. D. Hamilton. Analysis of time series subject to changes in regime. *Journal of econometrics*, 45(1):39–70, 1990.
- [62] J. D. Hamilton. Specification testing in markov-switching time-series models. *Journal of Econometrics*, 70(1):127–157, 1996.
- [63] D. M. Hawkins. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004.
- [64] M. Herbster and M. K. Warmuth. Tracking the best expert. *Machine Learning*, 32(2):151–178, 1998.
- [65] M. Herbster and M. K. Warmuth. Tracking the best expert. *Machine Learning*, 32:151–178, 1998.
- [66] J. R. Hershey and P. A. Olsen. Approximating the kullback leibler divergence between gaussian mixture models. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–317. IEEE, 2007.
- [67] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.
- [68] R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge University Press, 1985.
- [69] J. D. Hoyt and H. Wechsler. Detection of human speech in structured noise. In *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on*, volume 2, pages II–237. IEEE, 1994.

- [70] J. Hull, S. Treepongkaruna, D. Colwell, R. Heaney, and D. Pitt. *Fundamentals of futures and options markets*. Pearson Higher Education AU, 2013.
- [71] J. C. Hull. *Options, Futures, and Other Derivatives*. Prentice Hall, 6th edition, 2006.
- [72] N. Kanopoulos, N. Vasanthavada, and R. L. Baker. Design of an image edge detection filter using the sobel operator. *Solid-State Circuits, IEEE Journal of*, 23(2):358–367, 1988.
- [73] M. G. Kendall and J. K. Ord. *Time-series*, volume 296. Edward Arnold London, 1990.
- [74] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM SIGMOD Record*, 30(2):151–162, 2001.
- [75] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 289–296. IEEE, 2001.
- [76] E. Keogh, S. Chu, D. Hart, and M. Pazzani. Segmenting time series: A survey and novel approach. *Data mining in time series databases*, 57:1–22, 2004.
- [77] E. J. Keogh and M. J. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *KDD*, volume 98, pages 239–243, 1998.
- [78] E. J. Keogh and M. J. Pazzani. A simple dimensionality reduction technique for fast similarity search in large time series databases. In *Knowledge Discovery and Data Mining. Current Issues and New Applications*, pages 122–133. Springer, 2000.

- [79] C.-J. Kim. Unobserved-component time series models with markov-switching heteroscedasticity: Changes in regime and the link between inflation rates and inflation uncertainty. *Journal of Business & Economic Statistics*, 11(3):341–349, 1993.
- [80] R. C. Klemkosky and B. G. Resnick. Put-call parity and market efficiency. *The Journal of Finance*, 34(5):1141–1155, 1979.
- [81] E. Konstantinidi, G. Skiadopoulos, and E. Tzagkaraki. Can the evolution of implied volatility be forecasted? Evidence from European and US implied volatility indices. *Journal of Banking & Finance*, 32(11):2401–2411, 2008.
- [82] W. M. Koolen. Combining strategies efficiently: high-quality decisions from conflicting advice. 2011.
- [83] M. LeBlanc and R. Tibshirani. Combining estimates in regression and classification. *Journal of the American Statistical Association*, 91(436):pp. 1641–1650, 1996.
- [84] M. Levy and M. Sandler. New methods in structural segmentation of musical audio. In *Proceedings of the European Signal Processing Conference (EUSIPCO), Florence, Italy, 2006*.
- [85] M. Levy and M. Sandler. Structural segmentation of musical audio by constrained clustering. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(2):318–326, 2008.
- [86] M. Levy, M. B. Sandler, and M. A. Casey. Extraction of high-level musical structure from audio data and its application to thumbnail generation. In *ICASSP (5)*, pages 13–16. Citeseer, 2006.
- [87] J. L. E. K. S. Lonardi and P. Patel. Finding motifs in time series. In *Proc. of the 2nd Workshop on Temporal Data Mining*, pages 53–68, 2002.

- [88] M. Lovrić, M. Milanović, and M. Stamenković. Algorithmic methods for segmentation of time series: An overview.
- [89] L. Lu, H. Jiang, and H. Zhang. A robust audio classification and segmentation method. In *Proceedings of the ninth ACM international conference on Multimedia*, pages 203–211. ACM, 2001.
- [90] S. Lundbergh and T. Teräsvirta. *Forecasting with smooth transition autoregressive models*. Wiley Online Library, 2002.
- [91] H. Lutkepöhl. *New Introduction to Multiple Time Series Analysis*. Springer, 2005.
- [92] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, 1980.
- [93] A. Martin, D. Charlet, and L. Mauuary. Robust speech/non-speech detection using lda applied to mfcc. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 1, pages 237–240. IEEE, 2001.
- [94] J. Matejka, T. Grossman, and G. Fitzmaurice. Swifter: Improved online video scrubbing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 1159–1168, New York, NY, USA, 2013. ACM.
- [95] R. Mehrotra, K. R. Namuduri, and N. Ranganathan. Gabor filter-based edge detection. *Pattern Recognition*, 25(12):1479–1494, 1992.
- [96] R. C. Merton. Option pricing when underlying stock returns are discontinuous. *Journal of financial economics*, 3(1):125–144, 1976.
- [97] T. C. Mills and R. N. Markellos. *The econometric modelling of financial time series*. Cambridge University Press, 2008.

- [98] N. Mirghafori and C. Wooters. Nuts and flakes: A study of data characteristics in speaker diarization. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 1, pages I–I. IEEE, 2006.
- [99] M. T. Musavi, W. Ahmed, K. H. Chan, K. B. Faris, and D. M. Hummels. On the training of radial basis function classifiers. *Neural networks*, 5(4):595–603, 1992.
- [100] H. Nyquist. Certain topics in telegraph transmission theory. *American Institute of Electrical Engineers, Transactions of the*, 47(2):617–644, 1928.
- [101] E. Peiszer, T. Lidy, and A. Rauber. Automatic audio segmentation: Segment boundary and structure detection in popular music. *Proc. of LSAS*, 2008.
- [102] R. S. Pindyck and D. L. Rubinfeld. *Econometric models and economic forecasts*, volume 4. Irwin/McGraw-Hill Boston, 1998.
- [103] R. Plomp and W. J. Levelt. Tonal consonance and critical bandwidth. *The journal of the Acoustical Society of America*, 38(4):548–560, 1965.
- [104] T. Plotz, G. A. Fink, P. Husemann, S. Kanies, K. Lienemann, T. Marschall, M. Martin, L. Schillingmann, M. Steinrucken, and H. Sudek. Automatic detection of song changes in music mixes using stochastic models. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 665–668. IEEE, 2006.
- [105] R. E. Quandt. Tests of the hypothesis that a linear regression system obeys two separate regimes. *Journal of the American statistical Association*, 55(290):324–330, 1960.
- [106] R. E. Quandt. A new approach to estimating switching regressions. *Journal of the American statistical association*, 67(338):306–310, 1972.

- [107] D. A. Reynolds and P. Torres-Carrasquillo. The mit lincoln laboratory rt-04f diarization systems: applications to broadcast audio and telephone conversations. Technical report, DTIC Document, 2004.
- [108] C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *(ICML-1998) Proceedings of the 15th International Conference on Machine Learning*, pages 515–521. Morgan Kaufmann, 1998.
- [109] T. Scarfe, W. M. Koolen, and Y. Kalnishkan. A long-range self-similarity approach to segmenting dj mixed music streams. In *Artificial Intelligence Applications and Innovations*, pages 235–244. Springer, 2013.
- [110] B. Scholkopf, K.-K. Sung, C. J. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classifiers. *Signal Processing, IEEE Transactions on*, 45(11):2758–2765, 1997.
- [111] S. L. Sclove. Time-series segmentation: A model and a method. *Information Sciences*, 29(1):7–25, 1983.
- [112] M. Seeger. Low rank updates for the cholesky decomposition. *University of California at Berkeley, Tech. Rep*, 2007.
- [113] W.-H. Shin, B.-S. Lee, Y.-K. Lee, and J.-S. Lee. Speech/non-speech classification using multiple features for robust endpoint detection. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 3, pages 1399–1402. IEEE, 2000.
- [114] M. A. Siegler, U. Jain, B. Raj, and R. M. Stern. Automatic segmentation, classification and clustering of broadcast news audio. In *Proc. DARPA speech recognition workshop*, volume 1997, 1997.

- [115] N. Sugiura and R. Ogden. Testing change-points with linear trend. *Communications in Statistics-Simulation and Computation*, 23(2):287–322, 1994.
- [116] J. A. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, J. Suykens, and T. Van Gestel. *Least squares support vector machines*, volume 4. World Scientific, 2002.
- [117] Y. K. Tim Scarfe, Wouter M. Koolen. Segmentation of electronic dance music. *International Journal of Engineering Intelligent Systems for Electrical Engineering and Communications*, 2014.
- [118] S. E. Tranter and D. A. Reynolds. An overview of automatic speaker diarization systems. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(5):1557–1565, 2006.
- [119] M. Tsukamoto, S. Ogawa, Y. Natsuda, Y. Minowa, and S. Nishimura. Advanced technology to identify harmonics characteristics and results of measuring. In *Harmonics and Quality of Power, 2000. Proceedings. Ninth International Conference on*, volume 1, pages 341–346. IEEE, 2000.
- [120] G. Tzanetakis and F. Cook. A framework for audio analysis based on classification and temporal segmentation. In *EUROMICRO Conference, 1999. Proceedings. 25th*, volume 2, pages 61–67. IEEE, 1999.
- [121] G. Tzanetakis and P. Cook. Multifeature audio segmentation for browsing and annotation. In *Applications of Signal Processing to Audio and Acoustics, 1999 IEEE Workshop on*, pages 103–106. IEEE, 1999.
- [122] M. C. Vanier and J. M. Bower. A comparative survey of automated parameter-search methods for compartmental neural models. *Journal of computational neuroscience*, 7(2):149–171, 1999.
- [123] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

- [124] V. N. Vapnik and V. Vapnik. *Statistical learning theory*, volume 2. Wiley New York, 1998.
- [125] V. Vovk. Competitive on-line linear regression. *Advances in Neural Information Processing Systems*, pages 364–370, 1998.
- [126] V. Vovk. A game of prediction with expert advice. *Journal of Computer and System Sciences*, 56:153–173, 1998.
- [127] V. Vovk. Derandomizing stochastic prediction strategies. *Machine Learning*, 35(3):247–282, 1999.
- [128] V. Vovk. Competitive on-line statistics. *International Statistical Review*, 69(2):213–248, 2001.
- [129] V. Vovk. Kernel ridge regression. In *Empirical Inference*, pages 105–116. Springer, 2013.
- [130] V. Vovk and A. Gammerman. Complexity approximation principle. *The Computer Journal*, 42(4):318–322, 1999.
- [131] V. Vovk and F. Zhdanov. Prediction with expert advice for the brier game. *Journal of Machine Learning Research*, 10:2445–2471, 2009.
- [132] W. W.-S. Wei. *Time series analysis*. Addison-Wesley publ, 1994.
- [133] P. Wilmott. *Paul Wilmott introduces quantitative finance*. Wiley, 2nd edition, 2007.
- [134] Z. Yuan and Y. Yang. Combining linear regression models. *Journal of the American Statistical Association*, 100(472), 2005.
- [135] F. Zhdanov and Y. Kalnishkan. An identity for kernel ridge regression. *Theoretical Computer Science*, 473:157–178, 2013.

# Appendices

## Appendix A

# Genetic Search Figures

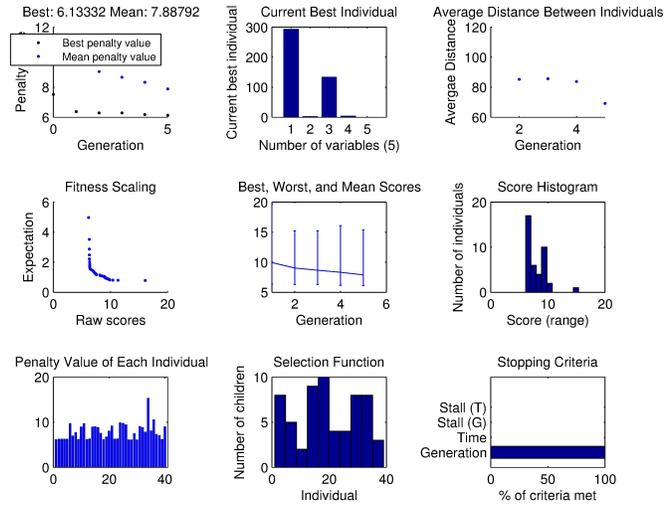


Figure A.1: Genetic parameter search for merging fixed regions (fixed-share).

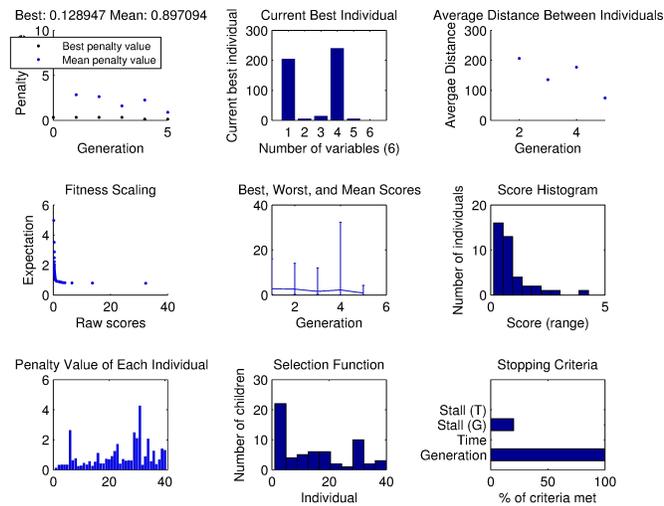


Figure A.2: Genetic parameter search for merging lagged regions (fixed-share).

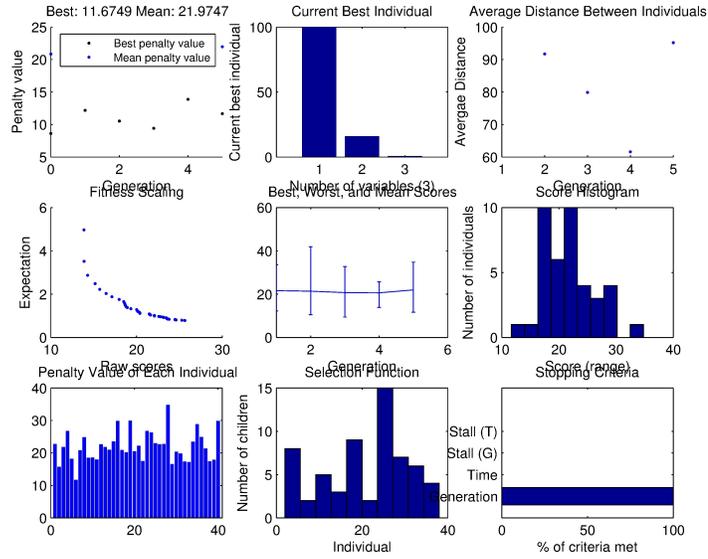


Figure A.3: Genetic parameter search for random model meta-merging (fixed-share)

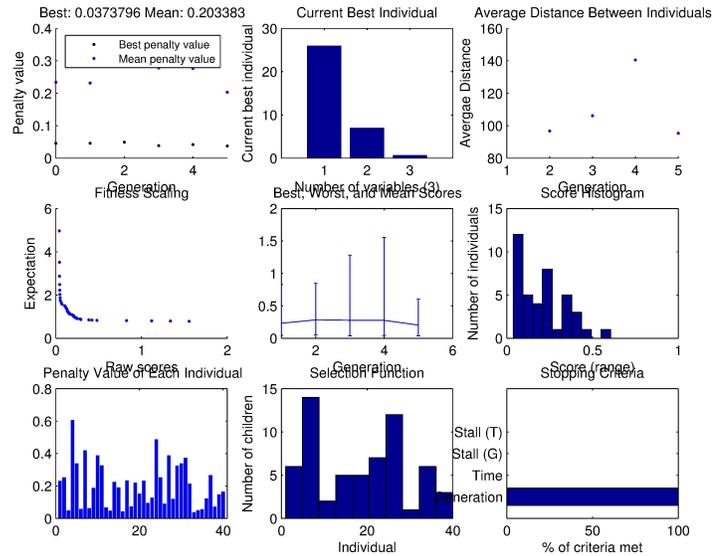


Figure A.4: Genetic parameter search for merged random ridge models, (fixed share), see Section 6.2.4.2.

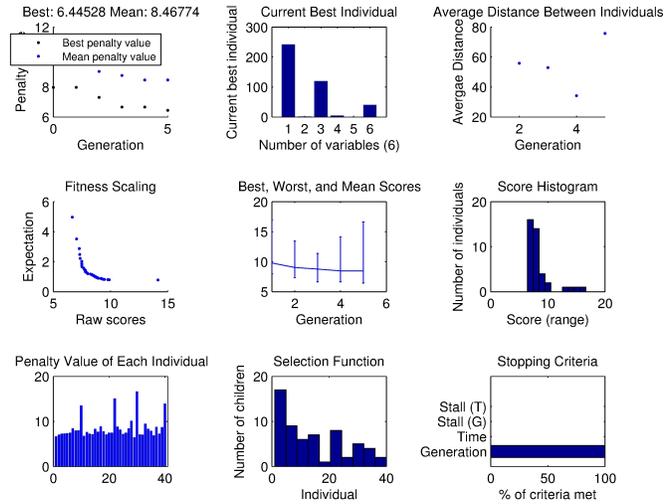


Figure A.5: Genetic parameter search for variable windows (fixed-share)

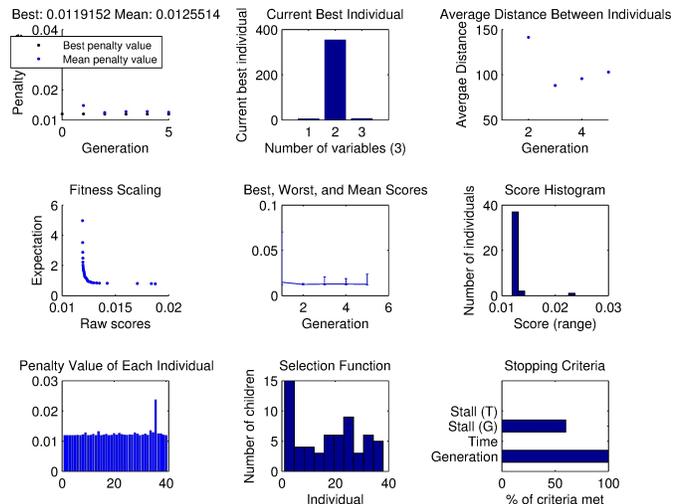


Figure A.6: Genetic parameter search for sliding window ridge regression.

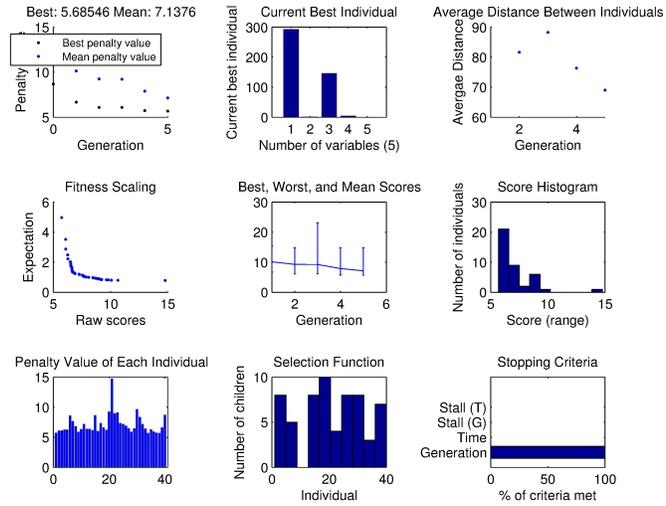


Figure A.7: Genetic parameter search for merging fixed regions (sleeping experts).

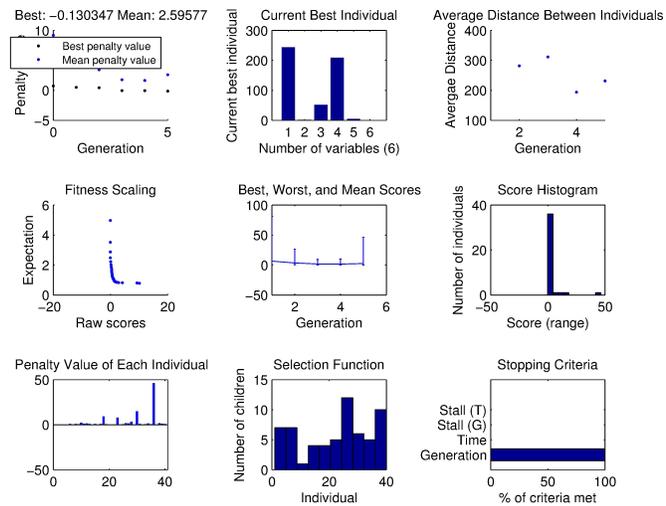


Figure A.8: Genetic parameter search for merging lagged regions (sleeping experts).

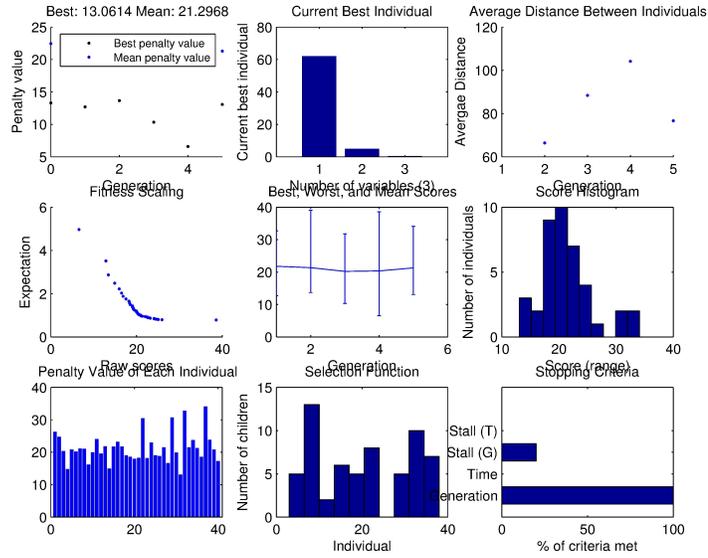


Figure A.9: Genetic parameter search for random model meta-merging (sleeping experts)

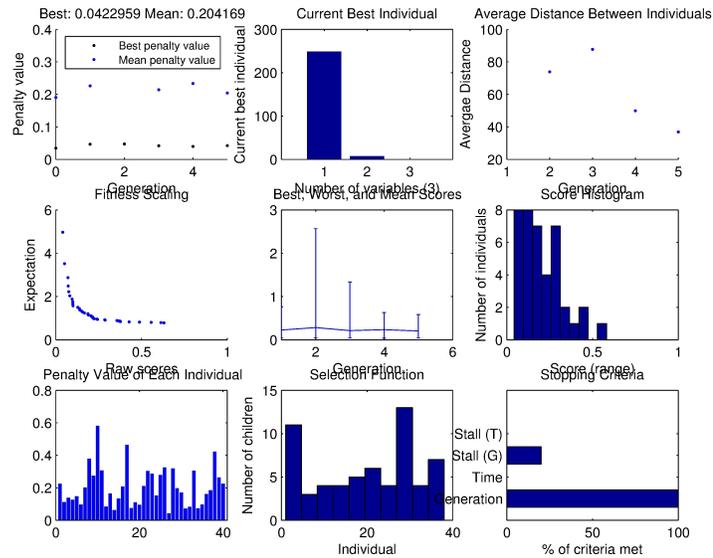


Figure A.10: Genetic parameter search for merged random ridge models, (sleeping experts), see Section 6.2.4.2.

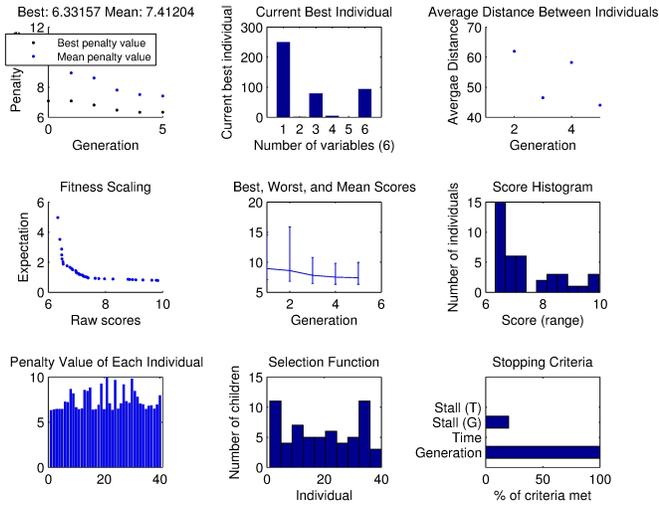


Figure A.11: Genetic parameter search for variable windows (sleeping experts)

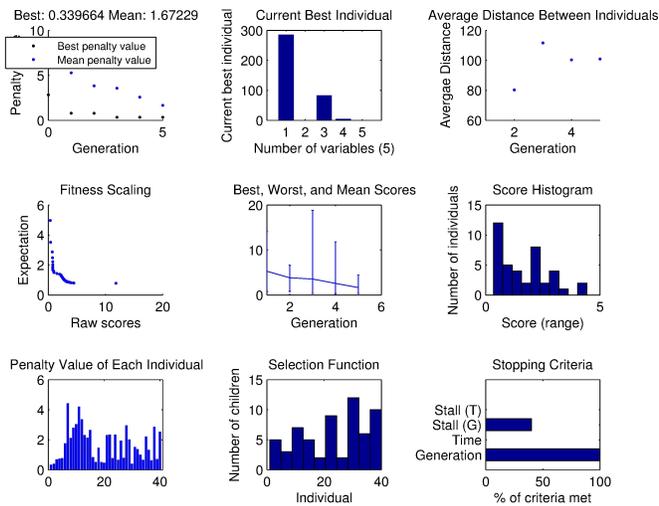


Figure A.12: Genetic parameter search for merging fixed regions (variable-share).

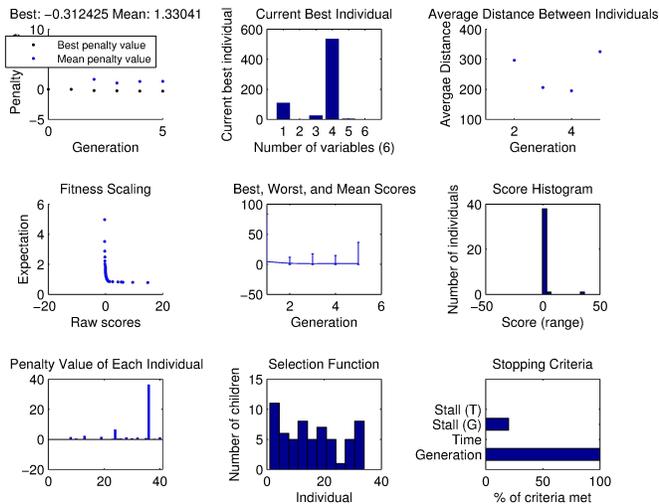


Figure A.13: Genetic parameter search for merging lagged regions (variable share).

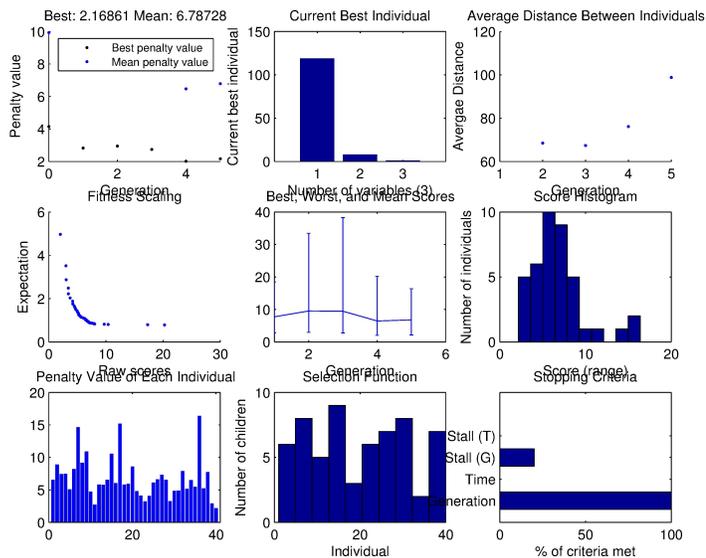


Figure A.14: Genetic parameter search for random model meta-merging (variable-share)

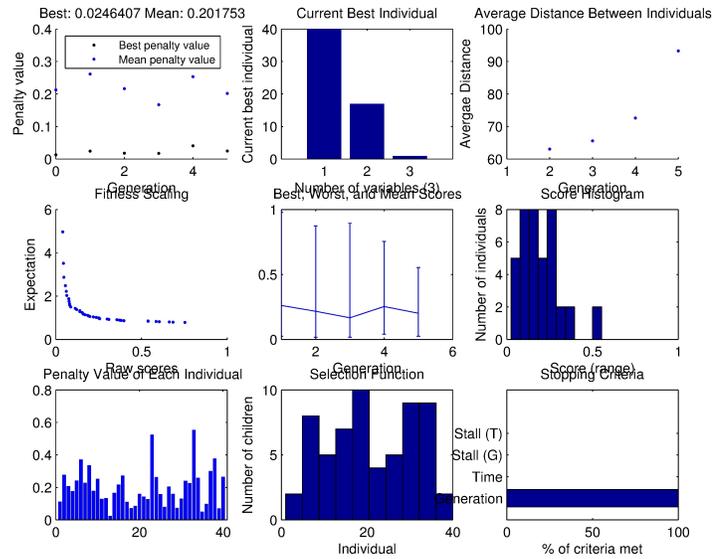


Figure A.15: Genetic parameter search for merged random ridge models, (variable share), see Section 6.2.4.2.

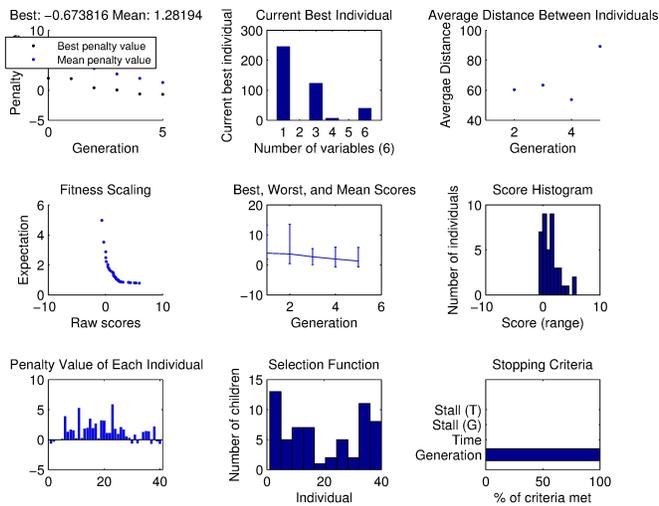


Figure A.16: Genetic parameter search for variable windows (variable-share)