Machine Learning for Financial Market Prediction

Tristan Fletcher

PhD Thesis

Computer Science
University College London

Declaration

I, Tristan Fletcher, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

The usage of machine learning techniques for the prediction of financial time series is investigated. Both discriminative and generative methods are considered and compared to more standard financial prediction techniques. Generative methods such as Switching Autoregressive Hidden Markov and changepoint models are found to be unsuccessful at predicting daily and minutely prices from a wide range of asset classes. Committees of discriminative techniques (Support Vector Machines (SVM), Relevance Vector Machines and Neural Networks) are found to perform well when incorporating sophisticated exogenous financial information in order to predict daily FX carry basket returns.

The higher dimensionality that Electronic Communication Networks make available through order book data is transformed into simple features. These volume-based features, along with other price-based ones motivated by common trading rules, are used by Multiple Kernel Learning (MKL) to classify the direction of price movement for a currency over a range of time horizons. Outperformance relative to both individual SVM and benchmarks is found, along with an indication of which features are the most informative for financial prediction tasks.

Fisher kernels based on three popular market microstructural models are added to the MKL set. Two subsets of this full set, constructed from the most frequently selected and highest performing individual kernels are also investigated. Furthermore, kernel learning is employed - optimising hyperparameter and Fisher feature parameters with the aim of improving predictive performance. Significant improvements in out-of-sample predictive accuracy relative to both individual SVM and standard MKL is found using these various novel enhancements to the MKL algorithm.

Acknowledgements

I would like to thank John Shawe-Taylor, David Barber, Zakria Hussain, Tom Diethe, Massimiliano Pontil, Pierre Mulotte, Rahul Savani, Erif Rison and Luke Chapman for their assistance with this PhD. I would also like to thank Anna Pamphilon and the rest of my family, Siddharth Saxena and Ollie Williams for their encouragement in starting and then pursuing this work and the support of my employers AtMet Capital, UBS and Aspect Capital.

The majority of this work would not have been possible without the data that EBS (ICAP) and the Chicago Mercantile Exchange (CME) kindly agreed to supply me with for research purposes.

Contents

List of Tables

List of Figures

1 Introduction

1.1 Summary

Time series relating to financial markets, for example daily prices of a share, are both highly non-stationary and noisy. Whether predictions can be made about future values of financial time series is of significant interest, not least because of the rewards such an ability would entail.

Another feature of financial time series is that in many cases they are multi-dimensional, so that for example instead of just having a scalar time series of prices for a currency one might have a ten dimensional vector time series representing prices of yet to be executed trades for each instance of the time series.

The non-stationary, noisy yet potentially multi-dimensional nature of financial time series makes them a suitable target for machine learning techniques.

This thesis considers both generative models - where instances of the time series are interpreted as significantly noisy observations of an underlying signal or latent state transition - and discriminative methods such as Support Vector Machines.

The document also considers canonical market microstructural models, all of which are generative by nature, and incorporates them into the discriminative machine learning domain through Fisher kernels. It is this marriage of traditional, often empirically based, market microstructural models to the machine learning framework that represents the main aim of the research described here.

Limited effort has been made in this work to examine the consequence of applying the methods investigated here in the real world: the focus of the work concentrates more on machine learning and market microstructure theory than the issues relating to the practical implementation of the models developed.

Amongst other things, the research will seek to answer the following questions:

- Can we make predictions in financial markets using machine learning techniques?
- How do generative and discriminative machine learning methods compare with each other when it comes to dealing with financial time series?
- Can we improve upon existing machine learning techniques to make them more suitable for financial prediction tasks?
- What features of financial time series are most useful when making predictions?
- Does incorporating domain knowledge of the financial markets, for example market microstructure, improve our predictive ability?

1.2 Research contribution

The overall goal of this research is to investigate the usage of machine learning methods in financial market prediction tasks. I aim to achieve this through incorporating canonical financial models into the discriminative and generative machine learning framework. Whilst progressing towards this goal, several research contributions are made.

The following list summarises the main areas of novelty that this research makes innovations in, along with where they are first introduced in this document:

- Research into using Switching Autoregressive Hidden Markov Models for forecasting futures contracts' daily returns (Section 4.1).
- Research into using changepoint models for forecasting futures contracts' daily returns (Section 4.2).
- Research into using discriminative machine learning techniques for the prediction of FX carry basket daily returns (Section 5).
- Use of Multiple Kernel Learning techniques in financial market prediction (Section 6).
- Use of order book features in financial market prediction (Sections 6 and 7).
- Use of market microstructural models in financial market prediction (Section 8).
- Incorporation of market microstructural models into the machine learning framework (Section 8).
- Use of Fisher features for financial market prediction (Section 8).
- Learning kernel mapping hyperparameter values and market microstructure parameters through learning the kernel (Section 9).

1.3 Publications

Parts of this thesis have been published in the *Proceedings of the International Conference of Financial Engineering* [95] and *Journal of Machine Learning Research Proceedings* [1] and presented at the *NIPS 2010 Workshop: New Directions in Multiple Kernel Learning* as well as the *Workshop on Applications of Pattern Analysis (WAPA) 2010*. More recent work has been submitted to the *Journal of Quantitative Finance*.

2 Related work / literature review

2.1 Econometrics

The models that econometricians have used in an attempt to tackle the problem of financial time series prediction have mostly attempted to capitalise on the fact that there is persistence (positive autocorrelation) in price movement - i.e. that when prices have been going up, they are likely to continue to go up and vice versa. A popular model which captures the trending nature of returns v_t is the Autoregressive Moving Average (ARMA) model of Box and Jenkins (1970) [2]:

$$v_t = \sum_{i=1}^{p} \alpha_i v_{t-i} + \sum_{i=1}^{q} \beta_i \epsilon_{t-i} + \epsilon_t + \delta$$

where the error term ϵ_t is often assumed to be an i.i.d. random variable sampled from a normal distribution, i.e. $\epsilon_t \sim N(0, \sigma^2)$.

The variance of these error terms is often modelled through Engle's (1982) Autoregressive Conditional Heteroskedasticity (ARCH) model [3]:

$$\sigma_t^2 = \sum_{i=1}^{p} \alpha_i \epsilon_{t-i}^2 + \delta$$

or its extension to the Generalised Autoregressive Conditional Heteroskedasticity (GARCH) model by Bollerslev (1986) [4]:

$$\sigma_t^2 = \sum_{i=1}^{p} \alpha_i \epsilon_{t-i}^2 + \sum_{i=1}^{q} \beta_i \sigma_{t-i}^2 + \delta$$

The main references describing the entire family of techniques are Hamilton (1994) [5], Alexander (2001) [6] and Lo & MacKinlay (2002) [7].

In terms of literature in this area, the initial work on financial market persistence is in stocks. Fama (1970) [8] found that the majority of stocks in the Dow Jones Industrial Average exhibited positive daily serial correlation. Fama and French (1988) [9] discovered that autocorrelations of portfolios of stocks form a U-shaped pattern across increasing return horizons with positive autocorrelations in the nearer term, reaching minimum (negative) values for 3-5 year returns and

then moving back toward zero for longer return horizons.

Using weekly data, Lo and MacKinlay (1988) [10] found significant positive autocorrelation for weekly and monthly holding-period index returns, but negative autocorrelations for individual securities. They later found negative autocorrelation in the weekly returns of individual stock returns, whilst weekly portfolio returns were strongly positively autocorrelated [11]. More recently, they found a positive autocorrelation for weekly holding-period market indices returns but a random walk for monthly ones [7].

Jegadeesh (1990) [12] found highly significant negative autocorrelation in monthly individual stock returns, but strong positive autocorrelation at twelve months. Campbell *et al.* (1996) [13] found that the autocorrelation of weekly stock returns is weakly negative, whilst the autocorrelations of daily, weekly and monthly stock index returns are positive. Ahn *et al.* (2002) [14] looked at daily autocorrelations of stock indices and found that they are positive even though the futures based on them are close to zero.

The characterisation of persistence in FX markets is more recent and described over much shorter time periods. Zhou (1996) [15] found that high-frequency FX data have extremely high negative first-order autocorrelation in their returns. This is backed up by Cont (2001) [16] who shows negative autocorrelation on a tick-by-tick basis for USDJPY and Dacorogna *et al.* (2001) [17], who highlight the negative autocorrelation of one minute FX returns.

2.2 Technical analysis

When the forecast horizon of a prediction represents medium to long term timescales, for example from days to months, methods focusing on fundamental macroeconomic trends are best placed to make predictions on how a price might evolve. These techniques often take the form of technical analysis to either make the outright predictions themselves or more commonly to ascertain turning points in these trends and hence the most suitable times to enter into or exit from trades.

Brown and Jennings (1989) [18] showed that technical analysis is useful when prices are not fully informative and traders have rational beliefs about the relation

between prices and signals.

Neftci (1991) [19] showed that a subset of the rules that technical analysts use generate successful forecasting techniques, but that even these well-defined rules were shown to be useless in prediction if the economic time series is Gaussian. However, if the processes under consideration are non-linear, the rules might capture some information.

Taylor and Allen (1992) [20] report the results of a survey among senior London-based foreign exchange dealers and found that the majority of them felt that technical analysis was an important tool and that there was a bias towards using technical as opposed to fundamental analysis for shorter time horizons.

Lui and Mole (1998) [21] report the results of another survey regarding the use of technical analysis amongst foreign exchange dealers in Hong Kong and indicate its use amongst the majority of respondents and its increased popularity for shorter time horizons.

Brock *et al.* (1992) [22] investigated 26 technical trading rules using 90 years of daily stock prices from the Dow Jones Industrial Average and found that they all outperformed the market. Neely *et al.* (1997) [23] used genetic programming to evolve technical trading rules in foreign exchange markets with great success. They later show evidence that using technical trading rules to trade during periods of US intervention in foreign exchange markets can be profitable over the short term. However, LeBaron (1999) [24] showed that after removing periods in which the Federal Reserve is active, exchange rate predictability is dramatically reduced using technical trading rules.

Lo *et al.* (2000) [25] explore the performance of technical analysis on US stocks from 1962 to 1996 and find that many technical indicators are informative.

Fernandez-Rodryguez *et al.* (2000) [26] use a technical trading rule-based neural network to forecast prices on the Madrid Stock Market and discover that a technical trading rule outperforms a simple buy-and-hold strategy for bear and stable markets but not bull ones. Neely and Weller (2001) [27] use genetic programming to evolve technical trading rules that are profitable during US foreign exchange

intervention.

Kavajecz and Odders-White (2004) [28] show that the technical analysis concepts of support and resistance levels coincide with peaks in depth on the limit order book and that moving average forecasts are informative regarding order book depth.

2.3 Market microstructure

Where econometrics and technical analysis concentrate more on the value of financial assets, market microstructure concentrates more on the actual trading process itself - analysing how specific mechanisms affect phenomena such as trading costs, prices, volume and trading behaviour in general. The field helps explain many of the costs that prevent assets from achieving their fundamental values.

Interest in market microstructure has grown substantially over the last two decades to reflect the rapid changes in the structure and technology of the world's financial markets. Structural shifts have arisen due to globalisation, the demutualisation of many exchanges and increasing competition between markets. Regulatory changes have also played a significant role. For example, governance from the Securities and Exchange Commission (SEC) in the US led to the creation of Electronic Communication Networks (ECNs), which have aggressively competed with exchanges for market share and data from which form the bulk of this research.

A summary of the empirical findings that have been gleaned from the following literature review can be found in Section 17.1 of the appendix.

2.3.1 Order books

The majority of trading for many financial products takes place on ECNs; this is most notable for currencies [29]. Continuous trading takes place on these exchanges via the arrival of market and limit orders. The latter specify whether the party wishes to buy or sell, the amount (volume) desired, and the price at which the transaction will occur. While traders had previously been able to view the prices of the highest buy (best bid) and lowest sell orders (best ask), a relatively recent development in certain exchanges is the real-time revelation of the total volume of trades sitting on the ECN's order book at both these price levels and

also at price levels above the best ask and below the best bid. See Table 1 for an example of an order book.

Table 1: An example of an order book showing three levels on each side

	Price	Volume($M)
	1.4752	1
	1.4751	8
Best Ask	1.4750	3
	↕ *Spread*	
Best Bid	1.4749	15
	1.4748	5
	1.4747	9

Bollerslev and Domowitz (1993) [30] simulate the effect of varying order book depth and find a relationship between depth and the autocorrelation in the variance of transaction prices. They also find increases in the autocorrelation of spreads as order book depth increases.

Hamao and Hasbrouck (1995) [31] investigated the behaviour of intraday trades and quotes for individual stocks on the Tokyo Stock Exchange. They found that when market orders were converted into limit orders, execution is delayed but that they executed at improved prices, that an order that is held with an indicative quote has a more significant price impact than one that is immediately executed in full and that the market tends to move in the direction of market orders after they are executed. They later examined NYSE SuperDOT market and limit orders and found that limit orders placed at or better than the prevailing quote perform better than market orders.

Maslov and Mills (2001) [32] studied NASDAQ order book data and found that a large asymmetry in the number of limit orders placed between the bid and ask sides of the book was shown to result in short term price changes.

Bouchaud, Mezard and Potters (2002) [33] investigated the order books of three liquid stocks of the Paris Bourse and found that incoming limit order prices follow a power law around the current price with a diverging mean.

Zovko and Farmer (2002) [34] examined two million orders from the London Stock Exchange and discovered that the difference between the limit price and the best price available had a cumulative power law distribution and that these relative limit price levels are positively correlated with volatility.

Potters and Bouchaud (2003) [35], using NASDAQ order book data, found that incoming limit order prices revealed a very slowly decaying tail and that the life-time of a given order increases as one moves away from the bid-ask. They also found a logarithmic (rather than power-law) relationship between price response and volume on French and British stocks.

Hall and Hautsch (2004) [36], investigating the Australian Stock Exchange (ASX), found that buy-sell pressure is particularly influenced by recent market activity and order book depth and that traders appear to use order book information in order to infer behaviour relating to other market participants.

Using trades and quotes data for liquid stocks on the Paris stock market, Bouchaud *et al.* (2004) [37] found that market orders on the Paris stock market exhibited persistence whilst limit orders were mean reverting; with the interplay between the two processes resulting in equilibrium because the persistence in order signs is compensated for by anticorrelated fluctuations in transaction size and liquidity.

Weber and Rosenow (2005) [38] calculated the mean price impact of market orders on the Island ECN order book. They discovered that the price impact function is convex and increases much faster than the concave impact function between returns and limit order flow; the anticorrelation leads to an additional influx of limit orders as a reaction to price changes, which reduces the price impact of market orders.

2.3.2 General market microstructure

Research in market microstructure for stocks starts as early as 1976 when Garman [39] modelled both dealership and auction markets using a collection of market agents. Morse and Ushman (1983) [40] examined the effect of information announcements on spreads and found no significant changes in spreads surrounding quarterly earnings announcements, but did find significant increases in spreads on the day of large price changes.

Lease, Masulis and Page (1991) [41] examine the relationship between spreads and event date returns on equity offerings by NYSE listed firms and found significant biases towards negative returns and larger spreads on the offering day.

Allen and Gorton (1992) [42] explain that buyers wish to avoid trading with informed investors and, because they are generally able to choose the time at which they trade, will tend to cluster. This means that when liquidity buyers are not clustering, purchases are more likely to be by an informed trader than sales, so the price movement resulting from a purchase is larger than for a sale. As a result, profitable manipulation by uninformed investors may occur.

Huang and Stoll (1994) [43] developed econometric models of quote revisions and transaction returns and used them to highlight the importance of different microstructure theories as well as to make predictions. Their most significant conclusion was that the expected quote return was positively related to the deviation between the transaction price and the quote midpoint, while the expected transaction return is negatively related to the same variable.

Brennan and Subrahmanyam (1996) [44] investigated the relation between monthly stock returns and measures of liquidity obtained from intraday data. They found a significant return premium associated with transaction costs.

Amihud, Mendelson and Lauterbach (1997) [45] showed that improvements in market microstructure are valuable in that stocks listed on the Tel Aviv stock Exchange that were transferred to a more efficient trading method were subject to significant and permanent price increases. A similar phenomenon was found by MacKinnon and Nemiroff (1999) [46], who examined the effect of the move to decimalization on the Toronto Stock Exchange and found many benefits to investors, such as reduced spreads and transaction costs and increased trading activity.

More general overviews of market microstructural work can be found in [47],[48], [49], [50], [51], [52], [53], [54] and [55].

2.4 Evolutionary techniques

Many attempts have been made to carry out financial market prediction in a more empirical manner, where little or no assumptions are made about how the time-series of observations come about or the underlying process generating them. One popular way of doing this has been to use evolutionary techniques, where parameters of models or the models themselves are evolved using techniques similar to genetic processes in the biological world, encompassing functions such as crossover and mutation.

Neely, Weller and Dittmar (1997) [56] use Genetic Programming [57] techniques to evolve technical trading rules. They discover evidence of significant out-of-sample excess returns to those rules for each of six exchange rates over the period 1981-1995.

Allen and Karjalainen (1999) [58] use Genetic Algorithms [59] to derive technical trading rules for the S&P 500, but fail to beat a simple buy-and-hold strategy benchmark.

Kaboudan (2000) [60] used Genetic Programming techniques to predict the price of six equities. He discovered that predicting actual prices is easier than predicting returns over a relatively small out-of-sample period of fifty trading days.

Potvin, Soriano and Vallee (2004) [61] employ Genetic Programming as a way to automatically generate short-term trading rules on the equity markets and concluded that the performance of the rules is heavily contingent on whether the market was rising or falling.

A summary of the performance of evolutionary techniques by Park and Irwin (2004) [62] found that Genetic Programming worked well on FX markets, but performed poorly on equities and futures.

Fletcher (2007) [63] investigated a combination of Evolutionary and Neural Network techniques called NEAT (Stanley (2002) [64]) on predicting high frequency FX triangulation arbitrage opportunities and concluded that the technique was of similar success to a simpler Kalman Filter-based benchmark and did not warrant the significantly increased computational cost required to implement the method.

2.5 Generative machine learning

2.5.1 Kalman filtering

If one assumes that price action is the result of Gaussian noisy observations of a latent linearly evolving process, one can model the system using a Linear Dynamical System (LDS):

$$h_t = Ah_{t-1} + \eta_t^h \qquad \eta_t^h \sim N(0, \Sigma_H) \tag{2.1}$$

$$v_t = Bh_t + \eta_t^v \qquad \eta_t^v \sim N(0, \Sigma_V) \tag{2.2}$$

where h_t represents the underlying latent state at time t, A describes how it moves from one time step to the next, η_t^h is the process noise which is i.i.d. normally distributed with zero mean and covariance Σ_H. The observations v_t are projections of h_t through the observation matrix B and are also subject to i.i.d. normally distributed noise of zero mean and covariance Σ_V.

A common method, particularly in the signal processing domain, for performing inference in the LDS described in (2.1) and (2.2) is to use Kalman Filtering [65]. This recursive estimator derives a set of predict and update equations to model the latent state and observation based on minimising the mean-square error of the posterior state estimate $\hat{h}_t$:

Predict:

$$\hat{h}_t = A\hat{h}_{t-1} \tag{2.3}$$

$$P_t = AP_{t-1}A^T + \Sigma_H \tag{2.4}$$

Update:

$$K_t = P_tB^T(BP_tB^T + \Sigma_V)^{-1} \tag{2.5}$$

$$\hat{h}_t = \hat{h}_t + K_t(v_t - B\hat{h}_t) \tag{2.6}$$

$$P_t = (I - K_tB)P_t \tag{2.7}$$

Kalman Filtering is used extensively in financial markets, both in its basic form and also where various assumptions such as the linearity of the latent state transition are relaxed (Extended Kalman Filter and the Unscented Kalman filter, e.g.

[66]). More recent innovations in the area are centred on Ensemble Kalman filtering (e.g. [67]).

The most notable usage in highly liquid markets is that of Bolland and Connor (1996) [68] & [69]. Fletcher (2007) [63] also used the technique in high frequency FX prediction.

2.5.2 Hidden Markov Models

An L^{th} order Markov process is one where the conditional independence $p(h_t|h_{1:t}) = p(h_t|h_{t-L:t-1})$ holds, e.g. for the most common case where $L = 1$, $p(h_t)$ depends only on the previous time step $p(h_{t-1})$. A Hidden Markov Model (HMM) is one where latent states h_t follow a Markov process and visible observations v_t are generated stochastically through an emission process which defines how observations are generated given the latent state, i.e. $p(v_t|h_t)$.

The joint distribution of the latent states and observations for T time steps can be expressed:

$$p(h_{1:T}, v_{1:T}) = p(v_1|h_1)p(h_1)\prod_{t=2}^{T} p(v_t|h_t)p(h_t|h_{t-1}) \tag{2.8}$$

The joint probability of the state at time t and the observations up to and including that time can be described recursively in a process known as the *Forward Recursion*:

$$p(h_t, v_{1:t}) = p(v_t|h_t)\sum_{h_{t-1}} p(h_t|h_{t-1})p(h_{t-1}, v_{1:t-1}) = \alpha(h_t) \tag{2.9}$$

The probability of all the observations from a time t to T given the previous state can also be derived recursively in a process known as the *Backward Recursion*:

$$p(v_{t:T}|h_{t-1}) = \sum_{h_t} p(v_t|h_t)p(h_t|h_{t-1})p(v_{t+1:T}|h_t) = \beta(h_{t-1}) \tag{2.10}$$

(2.9) and (2.10) can then be combined to give the joint probability of each latent state given the full set of T observations. This is called the *Forward-Backward*

Algorithm:

$$p(h_t|v_{1:T}) = \frac{\alpha(h_t)\beta(h_t)}{\sum_{h_t}\alpha(h_t)\beta(h_t)} \tag{2.11}$$

It is worth noting that HMM and LDS are in fact equivalent, with the former having a discrete state space representation and the latter a continuous. HMM appear a plausible way of modelling financial time series in that the market is often described as being in different regimes (latent states), which drive observable price movements. There has been much work in using these techniques in the financial market domain, e.g. [70], [71], [72], [73], [74], [75], [76], [77] and [78].

2.6 Discriminative machine learning methods

Financial market prediction problems can be expressed as an attempt to find a relationship between an output y and a set of D inputs $\boldsymbol{x}$ where $\boldsymbol{x} = \{x_1, x_2 \dots x_D\}$, i.e. $y = f(\boldsymbol{x})$. If y represents a future asset return or price observation at some point in the future, the function f could be learnt from in-sample training data so that when new unseen (out-of-sample) data is presented, a new prediction can be made. Both regression where $y \in \Re$ and classification where $y \in \{-1, +1\}$, e.g. a return is positive or negative, would be useful to investigate.

$\boldsymbol{x}$ could be composed of exogenous variables or L lags of y, T time steps into the future so that:

$$\begin{aligned} y_{t+T} &= f(\boldsymbol{x}_t) \\ \text{where } \boldsymbol{x}_t &= \left\{x_t^1 \dots x_t^D, y_t, y_{t-1} \dots y_{t-L}\right\} \end{aligned} \tag{2.12}$$

2.6.1 Neural Networks

A popular, more recent method for making predictions in financial markets, sometimes incorporating technical analysis, is that of Artificial Neural Networks (ANN). ANN embody a set of thresholding functions connected to each other with adaptive weights that are trained on historical data in order that they may be used to make predictions in the future, see for example [79], [80], [81] and [82].

These techniques are often criticised for the stochastic nature of their weight initialisation, the fact that they cannot be guaranteed to provide optimal solutions

(they fail to converge on global optima) and that they are prone to overfitting. A more novel method that is not subject to these criticisms, but that is nevertheless well placed to deal with the high dimensional data sets that order books reveal is Support Vector Machines (SVM) [83].

2.6.2 Support Vector Machines

Cortes and Vapnik's (1995) Support Vector Machine [83] represent the relationship between an output y and a set of inputs $\boldsymbol{x}$ in the form:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} w_i \phi(\boldsymbol{x}) + b \tag{2.13}$$

where $\phi(\boldsymbol{x})$ represents a non-linear mapping of $\boldsymbol{x}$ into a higher dimensional feature space, i.e. a basis function, and $\boldsymbol{w}$ and b are parameters learnt from the N instances of training data.

In classification, these parameters are found by using Quadratic Programming (QP) optimisation to first find the α_i which maximize:

$$\begin{aligned} &\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j}^{N} \alpha_i \alpha_j y_i y_j \phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_j) \\ \text{where} \quad &\alpha_i \geq 0 \ \forall_i, \ \sum_{i=1}^{N} \alpha_i y_i = 0 \end{aligned} \tag{2.14}$$

The α_i are then used to find $\boldsymbol{w}$:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \phi(\mathbf{x}_i) \tag{2.15}$$

The set of Support Vectors $\mathcal{S}$ is then found by finding the indices i where $\alpha_i > 0$. b can then be calculated:

$$b = \frac{1}{N_s} \sum_{s \in \mathcal{S}} \left(y_s - \sum_{m \in S} \alpha_m y_m \phi(\boldsymbol{x}_m) \cdot \phi(\boldsymbol{x}_s) \right) \tag{2.16}$$

The mapping $\boldsymbol{x} \rightarrow \phi(\boldsymbol{x})$ is intended to make the data linearly separable in the feature space, and to this aim kernels $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_j)$ representing the

Radial Basis Function:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\left(\frac{\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2}{2\sigma^2}\right)}$$

and the Linear Kernel:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i \boldsymbol{x}_j^T$$

are commonly used.

For regression, one first needs to decide how significantly misclassifications should be treated (C) and how large the insensitive loss region inside which misclassifications are ignored should be (ϵ). One then proceeds by using QP optimisation to find the $\boldsymbol{\alpha}^+$ and $\boldsymbol{\alpha}^-$ which maximize:

$$\sum_{i=1}^{N}(\alpha_i^+ - \alpha_i^-)t_i - \epsilon\sum_{i=1}^{N}(\alpha_i^+ + \alpha_i^-) - \frac{1}{2}\sum_{i,j}(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-)\phi(\mathbf{x}_i)\cdot\phi(\boldsymbol{x}_j)$$

subject to the constraints ($\forall_i$):

$$\begin{aligned} 0 \leq \alpha_i^+ &\leq C \\ 0 \leq \alpha_i^- &\leq C \\ \sum_{i=1}^{N}(\alpha_i^+ - \alpha_i^-) &= 0 \end{aligned} \tag{2.17}$$

The α_i^+ and α_i^- are then used to find $\boldsymbol{w}$:

$$\mathbf{w} = \sum_{i=1}^{N}(\alpha_i^+ - \alpha_i^-)\phi(\boldsymbol{x}_i) \tag{2.18}$$

The set of Support Vectors $\mathcal{S}$ is then obtained by finding the indices i where $0 < \alpha_i < C$ and $\xi_i = 0$. b can then be calculated:

$$b = \frac{1}{N_s}\sum_{s\in\mathcal{S}}\left(t_i - \epsilon - \sum_{m=1}^{L}(\alpha_i^+ - \alpha_i^-)\phi(\boldsymbol{x}_i)\cdot\phi(\boldsymbol{x}_m)\right) \tag{2.19}$$

There has been much work in using SVM and other similar single-kernel based methods to predict the movement of financial time series, e.g. [84], [85], [86], [87],

[88], [89], [90], [91], [92], [93], [94], [95] and most notably [96].

2.6.3 Relevance Vector Machines

Tipping's Relevance Vector Machine (2001) [97] implements a Bayesian probabilistic methodology for learning in models of the form shown in (2.13). A prior is introduced over the model weights governed by a set of hyperparameters, one associated with each weight (α_i), whose most probable values are iteratively estimated from the data. If one assumes that the N target values $\boldsymbol{t}$ that one is attempting to predict are samples from the model subject to Gaussian distributed noise of zero mean and variance σ^2, and that both $\boldsymbol{\alpha}$ and σ^2 have uniform distributions, one can derive the model evidence:

$$\begin{aligned} p(\boldsymbol{t}|\boldsymbol{\alpha}, \sigma^2) &= \int p(\boldsymbol{t}|\boldsymbol{w}, \sigma^2) p(\boldsymbol{w}|\boldsymbol{\alpha}) d\boldsymbol{w} \\ &= \frac{1}{2\pi^{\frac{N}{2}}} \left|\sigma^2 \boldsymbol{I} + \boldsymbol{\Phi} \boldsymbol{A}^{-1} \boldsymbol{\Phi}^T\right|^{-\frac{1}{2}} \exp\left\{-\frac{\boldsymbol{t}^T}{2}(\sigma^2 \boldsymbol{I} + \boldsymbol{\Phi} \boldsymbol{A}^{-1} \boldsymbol{\Phi}^T)^{-1} \boldsymbol{t}\right\} \end{aligned}$$

where $\boldsymbol{A} = diag(\alpha_0, \alpha_1, \ldots, \alpha_N)$, $\boldsymbol{I}$ is the $N \times N$ identity matrix and $\boldsymbol{\Phi}$ is the $N \times D$ design matrix constructed such that the ith row represents the vector $\phi(\boldsymbol{x}_i)$.

This evidence can be maximized by Mackay's (1992) [98] evidence procedure:

1. Choose starting values for $\boldsymbol{\alpha}$ and β.
2. Calculate $\boldsymbol{m} = \beta \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \boldsymbol{t}$ and $\boldsymbol{\Sigma} = (\boldsymbol{A} + \beta \boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1}$ where $\beta = \sigma^{-2}$.
3. Update $\alpha_i = \frac{\gamma_i}{m_i^2}$ and $\beta = \frac{N - \sum_i \gamma_i}{\|\boldsymbol{t} - \boldsymbol{\Phi} \boldsymbol{m}\|^2}$.
4. Prune the α_i and corresponding basis functions where $\alpha_i >$ a threshold value (corresponding to w_i with zero mean).
5. Repeat (2) to (4) until a convergence criterion is met.

The hyperparameter values $\boldsymbol{\alpha}$ and β which result from the above procedure are those that maximize the marginal likelihood and hence are those used when making a new estimate of a target value t for a new input $\boldsymbol{x}'$:

$$t = \boldsymbol{m}^T \phi(\boldsymbol{x}') \tag{2.20}$$

The variance relating to the confidence in this estimate is given by:

$$\sigma^2(\boldsymbol{x}') = \beta^{-1} + \phi(\boldsymbol{x}')^T \boldsymbol{\Sigma} \phi(\boldsymbol{x}') \tag{2.21}$$

There is very limited evidence of research using RVM in the financial domain other than Gestel *et al.* (2001) [90], Hazarika (2002) [91], Tino & Yao (2005) [92] & Huang & Wu (2008) [94]. It is probable that the reason for the limited evidence of the use of RVM in the financial domain relative to SVM is that although the techniques have a similar functional form, the RVM's evidence procedure described above makes them not only more complex and computationally expensive to implement but also occasionally prone to over-fitting.

Fletcher (2009) [95] found the use of a committee of ANN, SVM and RVM techniques was superior to any known benchmark when predicting daily FX carry basket returns.

Despite the many advantages of using SVM/RVM and their resulting popularity, one of the main problems of the SVM approach for real-world problems is the selection of the feature-space mapping through the choice of kernel, which is often selected empirically with little theoretical justification.

3 Conclusions from literature review

The econometrics literature points to the fact that weekly and monthly stock returns are weakly negatively correlated, whilst daily, weekly and monthly index returns are positively correlated. This is very different in FX, where short term returns (i.e. of the order of a minute or less) are highly negatively correlated. The consistency of the models described and results reported in this area make it a very useful source of benchmarks for the techniques this research will develop.

The majority of technical analysis is concerned with incorporating previous price action of the time series in question, occasionally incorporating simple measures of the volume traded, for example to provide an estimate of information quality that cannot be deduced from the price [99] or to confirm trends [100] and [101]. A review of the field by Park (2004) [102] suggests that technical analysis works best on currency markets, intermediate on futures markets, and worst on stock markets. This review also suggests that the efficacy of technical analysis is in decline as markets become increasingly efficient through the decline in transaction costs, increased computing power and more sophisticated market participants. This decline is reflected in Neely (2009) [103] - an unusually recent example of work in a field that has increasingly fewer publications associated with it.

There is a great deal of research on order books and the related field of market microstructure, but it is heavily based on stocks and often relates to characterising features such as liquidity, volatility and spreads (the difference between the best bid and ask prices) instead of attempting to predict future price action. The exposure of order books' previously hidden depths that has occurred more recently should allow traders to capitalise on the greater dimensionality of data available to them at every order book update (tick) when making trading decisions and permit techniques that are more sophisticated than the standard time series analysis toolset to be used when forecasting prices. Published literature does not indicate that this has occurred.

The evidence on the efficacy of evolutionary algorithms in financial prediction applications is divided. Furthermore, much of the work appears highly empirical, involving tweaking algorithms through trial and error, and in the author's opinion does not warrant formal research. The author also has a preference for research

that is fully repeatable and does not involve the random/stochastic elements which form the basis of most evolutionary techniques.

There is a significant body of work implementing signal processing methods in financial time-series prediction. Similar to econometric techniques, many of the methods can be seen as special cases of the ideas I propose developing (e.g. the Kalman Filter is a recursive estimator for an LDS). However, the framework under which the prediction problems are posed is not consistent with the Bayesian inference or even machine learning methodology that my research is centred around and it will therefore be difficult to develop novel techniques as extensions from the signal processing domain.

The literature suggests that standard HMM techniques can be effective in very specific applications, for example when predicting daily FX returns. However, the standard HMM framework appears to be too simplistic to be useful for the majority of financial market prediction problems.

The stochastic nature of ANN's weight initialisation, the fact that they cannot be guaranteed to provide optimal solutions (they fail to converge on global optima) and that they are prone to overfitting make ANN less of a useful forecasting tool than SVM. However, despite the many advantages of using SVM and their resulting popularity, one of the main problems of the SVM approach for real-world problems is the selection of the feature-space mapping through the choice of kernel, which is often selected empirically with little theoretical justification.

4 Generative machine learning research

Given the scarcity of evidence of using generative machine learning methods in financial prediction tasks, it was felt that a contribution could be made by investigating non-standard methods from this area. The evidence of success of HMM in financial prediction tasks, coupled with the knowledge that financial returns are at least to some extent autocorrelated, made Switching Autoregressive Hidden Markov Models an appropriate starting point.

4.1 Switching Autoregressive Hidden Markov Models

4.1.1 Theoretical model

Ephraim and Roberts' (2005) [104] Switching Autoregressive Hidden Markov Model (SAR-HMM) is similar to the standard HMM described in (2.8) but instead of the observations of the time series v_t being generated by the hidden states h_t at any given time t, the latent states describe an autoregressive relationship between sequences of observations. The extension of HMM to SAR-HMM is potentially relevant in that markets often go through periods where they trend or mean-revert, corresponding to positive and negative autoregressive coefficients respectively.

The SAR-HMM switches between sets of autoregressive parameters with probabilities determined by a state transition probability similar to that of a standard HMM:

$$v_t = \sum_{r=1}^{R} a_r(h_t) v_{t-r} + \eta_t \text{with } \eta_t \sim N(0, \sigma^2) \tag{4.1}$$

where $a_r(h_t)$ is the r^{th} autoregressor when in state $h \in \{1 \dots k\}$ at time t and each η_t is an i.i.d. normally distributed innovation with mean 0 and variance σ^2.

4.1.2 Inference

The definition in (4.1) allows us to to describe the probability of an observation at time t expressed as a function of the previous R observations and the current state h_t:

$$p(v_t | v_{t-R:t-1}, h_t) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{ -\frac{1}{2\sigma^2} \left(v_t - \sum_{r=1}^{R} a_r(h_t) v_{t-r} \right)^2 \right\} \tag{4.2}$$

The forward recursion calculates at time t the probability of being in a state h_t given all the observations *up to that point in time* $v_{1:t}$, i.e. $p(h_t|v_{1:t})$.

Starting with an initial probability of being in a state given the first observation:

$$p(h_1|v_1) \propto p(v_1|h_1)p(h_1)$$

values for $p(h_t|v_{1:t})$ for subsequent values of t can be found by iteration:

$$p(h_t|v_{1:t}) = \sum_{h_{t-1}} p(v_t|v_{t-R:t-1}, h_t)p(h_t|h_{t-1})p(h_{t-1}|v_{1:t-1}) \tag{4.3}$$

where each $p(h_{t-1}|v_{1:t-1})$ is the result of (4.3) for the previous time step $t-1$ and $p(v_t|v_{t-R:t-1}, h_t)$ is calculated from (4.2). The full derivation of (4.3) is shown in Section 17.2.1 of the appendix.

The forward recursion can be started at $t = R + 1$ with the previous posterior equalling the initial starting probability for each state, i.e. $p(h_R|v_{1:R}) = p(h_1)$.

Once the forward recursion has been completed for all time steps $t = 1 : T$, the backward pass calculates at time t the probability of being in a state h_t given all the observations *in the entire sequence of* T *time steps* $v_{1:T}$, i.e. $p(h_t|v_{1:T})$.

Commencing at the final time step ($t = T$) and working backwards to the start ($t = 1$), $p(h_t|v_{1:T})$ can be evaluated as follows:

$$p(h_t|v_{1:T}) = \sum_{h_{t+1}} p(h_{t+1}|h_t)p(h_t|v_{1:t})p(h_{t+1}|v_{1:T}) \tag{4.4}$$

where each $p(h_{t+1}|v_{1:T})$ is the result of (4.4) from a previous iteration (i.e. future time step $t+1$) and $p(h_t|v_{1:t})$ for each time step has been calculated by the forward pass in (4.3). The full derivation of (4.4) is shown in Section 17.2.1 of the appendix.

4.1.3 Experiments

Experiments were conducted attempting to predict the daily returns for each of 44 futures contracts spanning 10.5 years (see Table 14 in Section 17.3 of the appendix for the futures contracts investigated). It was found that the performance

of the Switching Autoregressive HMM was highly sensitive to the choice of parameter values $\boldsymbol{\theta} = \{R, \boldsymbol{a}, p(h_1|v_1), p(h_{t+1}|h_t)\}$ and hence a parameter set that was appropriate for one time period would give very poor results for another. Expectation Maximisation was used in conjunction with K-folds cross-validation to optimise $\boldsymbol{\theta}$ for each fold, but this problem of starting point sensitivity was never overcome and experiments using this technique on daily returns were discontinued.

The work was repeated on attempting to predict minutely returns on each of the the 44 futures contracts for a subset of ten different days but exactly the same problems were encountered and the experiments abandoned.

4.2 Changepoint models

The failure of the SAR-HMM to effectively model financial returns motivated an investigation into a different class of generative model. In the same way that the SAR-HMM seemed plausible because markets often go through periods where they trend or mean-revert - these different states being represented by the alternative autoregressive parameters - they can also be thought of as occasionally going through structural changes in other aspects of the time series, for example the variance. A class of models that aims to characterise such sudden alterations in behaviour is that of changepoint models.

4.2.1 Theoretical model

Changepoint models - see for example [105] and [106] - assume one or more parameters defining the distribution of one or more observable variables are constant for a run of length r_t at time t, but change when there is a changepoint and a new run starts, i.e. $r_t = 0$.

Their use in this context would be to predict at time t the next observation x_{t+1} given the vector of all previous observations $\mathbf{x}_{1:t}$. One could assume, for example, that observations are Gaussian distributed, i.e. $P(x_t) \sim N(\mu_t, \sigma_t^2)$ and that the variance σ_t^2 (or its inverse, the precision λ_t) is sampled from a Gamma prior (i.e. $\lambda_t \sim Gam(\alpha_0, \beta_0)$) and the mean μ_t is sampled from a Gaussian (i.e. $\mu_t \sim N(\mu_0, (\kappa_0\lambda)^{-1})$), each time a changepoint occurs. Another assumption could be that the probability distribution of a run being of length r_t at time t given its length the previous time step is as follows:

$$P(r_t|r_{t-1}) = \begin{cases} P_{cp} & \text{if } r_t = 0 \\ 1 - P_{cp} & \text{if } r_t = r_{t-1} + 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

where in general P_{cp} will be small so that $P(r_t = 0|r_{t-1})$, i.e. the probability of a changepoint occurring at time t, will be low.

4.2.2 Implementation

See Section (17.2.2) in the appendix for a full derivation of the equations used in this section.

Assuming that we are starting at time $t = 1$ with a zero run length, i.e. that $r_0 = 0$ and that values for all the hyperparameters have been chosen, conjugacy of the priors allows inference in the model to proceed as follows for each time step:

- Assume that we are starting at time t with a zero run length, i.e. that $r_0 = 0$. Choose values for μ^0, ν^0 and λ^0.
- Make a new observation x_t.
- Evaluate Predictive Probabilities for all r_t where $0 < r_t < t$:
  $P(x_t|r_{t-1}, \mathbf{x}_{t-1}^{r_{t'}}) = St(x_t|\mu_t, \alpha_t, \sigma_t^2)$, i.e. x_t is distributed with a Student t-distribution with a mean μ_t, $2\alpha_t$ degrees of freedom and a variance of σ^2.
- Calculate Growth Probabilities for all r_t where $0 < r_t < t$:
  $P(r_t = r_{t-1} + 1, \mathbf{x}_{1:t}) = P(r_{t-1}, \mathbf{x}_{1:t-1})P(x_t|r_{t-1}, \mathbf{x}_{t-1}^{r_{t'}})(1 - P_{cp})$
- Calculate Changepoint Probabilities for all r_t where $0 < r_t < t$:
  $P(r_t = 0, \mathbf{x}_{1:t}) = \sum_{r_{t-1}} P(r_{t-1}, \mathbf{x}_{1:t-1})P(x_t|r_{t-1}, \mathbf{x}_{t-1}^{r_{t'}})P_{cp}$
- Calculate Evidence:
  $P(\mathbf{x}_{1:t}) = \sum_{r_t} P(r_t, \mathbf{x}_{1:t})$
- Determine Run Length Distributions for all r_t where $0 < r_t < t$:
  $P(r_t|\mathbf{x}_{1:t}) = \frac{P(r_t, \mathbf{x}_{1:t})}{P(\mathbf{x}_{1:t})}$
- Update all the parameters:

$$\begin{aligned}
\mu_{t+1} &= \frac{(\kappa_t \mu_t + x_t)}{\kappa_t + 1} \\
\alpha_{t+1} &= \alpha_t + 0.5 \\
\kappa_{t+1} &= \kappa_t + 1 \\
\beta_{t+1} &= \beta_t + \frac{\kappa_t (x_t - \mu_t)^2}{2(\kappa_t + 1)} \\
\sigma_{t+1}^2 &= \frac{\beta_t(\kappa_t + 1)}{\alpha_t \kappa_t}
\end{aligned}$$

- Make a prediction of what the next observation will be using these updated parameters:
  $P(x_{t+1}|\mathbf{x}_{1:t}) = \sum_{r_t} P(x_{t+1}|r_t, \mathbf{x}_t^{r_t})P(r_t|\mathbf{x}_{1:t})$

4.2.3 Experiments

Experiments were once again conducted attempting to predict the daily returns for each of 44 futures contracts spanning 10.5 years (2751 data points) and minutely returns on each of the the 44 futures contracts for a subset of ten different days, but similar problems as with the SAR-HMM were encountered and the experiments abandoned.

4.3 Conclusions

Though modelling price action as observations generated from some latent process seems plausible, particularly when incorporating either the known autocorrelation of asset returns or the sudden structural changes that occur in a market, the large number of parameters that the two generative models investigated here required rendered them impractical for financial prediction tasks. The low signal to noise ratio in asset price returns meant that when using models with large numbers of parameters overfitting frequently occurs and a set of parameters that was useful for one period seldom seems to be appropriate for the following one.

Furthermore, some of the mathematical assumptions that are used in formulating these two models, for example selecting exponential family distributions for the conjugate priors in the changepoint models in order to make the mathematics tractable, have no real-life financial motivation. For example, there is no reason that the priors used in Section 4.2.1 should be Gamma and Gaussian distributions, other than to make the maths tractable (the former being the conjugate prior of the latter). The choice of probability distributions which the generative models briefly investigated here consist of seem to have no foundation in financial theory - where it is known for example that returns from most asset classes follow a distribution with much higher kurtosis than the Gaussian. A useful innovation in this area could be to use models that have been shown to be more appropriate to the time series they are modelling but are also either still of the exponential family to ensure mathematical tractability or where sensible numerical approximations can be made so that conjugacy is not necessary. One might expect this significant weakness of the models investigated to also be the case for the majority of other standard generative machine learning methods used to model financial processes.

A greater effort could have been made in both attempting to make the methods investigated here successful. This could have been done in many ways, perhaps for example by extending the models to incorporate exogenous factors known to have an impact on the time series being predicted, incorporating regularisation into the models to reduce over-fitting or even to investigate more of the many alternative generative machine learning methods available. However, because of the weaknesses of these methods as described above, it was felt that researching discriminative methods would be an appropriate next step and that this area could be revisited - perhaps in conjunction with the discriminative framework.

5 FX carry basket prediction

Temporarily putting aside generative models, it seemed appropriate to investigate models that are not subject to the weaknesses outlined previously - namely potential over-parametrisation and the circumstantial choice of probability distribution. With this aim in mind, it was decided to research standard discriminative machine learning techniques incorporating sophisticated exogenous financial data.

It was expected that the novel inclusion of such data and the highly pragmatic and less theoretically motivated approach would be reflected in improved forecast ability.

The work in this section is published in the Proceedings of the International Conference of Financial Engineering 2009 [95].

5.1 Background

A portfolio consisting of a long position in one or more currency pairs with high interest rates and a short position in one or more currencies with low interest rates (an FX carry basket) is a common asset for fund managers and speculative traders. Profit is realized by owning (carrying) this basket owing to the difference in the interest rates between the high yielding currencies and the low yielding ones. The returns that this basket generate are subject to the risk that the difference between the yields might reduce, possibly becoming negative, and the fact that the exchange rates of the currencies might move unfavourably against the basket holder. A common basket composition is of the three highest yielding G10 currencies bought against the three lowest ones, updated daily to reflect any changes in yield rankings. This basket has a long-bias in the sense that someone holding it will tend to earn a positive return on the asset, subject to periods of negative returns (draw downs).

It would clearly be useful to an FX carry basket trader to be able to predict negative returns before they occur, so that the holder could sell out of or even go short the asset class before it would realize a loss. Several market-observable factors

are known to hold a strong relationship with FX carry returns. Furthermore, the returns are known to exhibit short term persistence (i.e. autocorrelation) [16]. It is upon these two phenomena that a trader may wish to capitalise, attempting to predict when returns will be negative and hence reduce the risk of realising poor returns for the asset over the holding period.

Artificial Neural Networks (ANN) have been used extensively in the general area of financial time-series prediction, with varying success e.g. [79], [107], [81] & [82] and hence represent a good starting point with the prediction problem posed here. Support Vector Machines (SVM) [83], being a more recent technique, have been used to a lesser extent e.g. [84], [86], [87] & [89] and indeed there is little evidence of their use in FX carry basket prediction and very little work on the incorporation of exogenous variables when making predictions. The more novel RVM has been used even less in the financial domain e.g. [90], [91], [92] & [93] and apparently never in the area of FX carry prediction.

Knowing of the relation between carry returns for any given day and both observable exogenous factors for that day and previous days' returns, it makes sense to attempt to incorporate these as inputs into a model where future returns are predicted given information available at the present.

The FX carry basket prediction problem can be expressed as attempting to find a relationship between an output y and a set of D inputs $\mathbf{x}$ where $x = \{x_1, x_2 \dots x_D\}$, i.e. $y = f(\mathbf{x})$, where y is used to represent a future return of the carry basket, either $T = 1$ or $T = 5$ days into the future. Both regression where $y \in \Re$ and classification where $y \in \{-1, +1\}$, i.e. the T-day return is positive or negative, are investigated. $\mathbf{x}$ is composed of five exogenous variables and L lags of y, so that:

$$\begin{aligned} y_{t+T} &= f(\boldsymbol{x}_t) \\ \text{where } \boldsymbol{x}_t &= \left\{x_t^1 \dots x_t^5, y_t, y_{t-1} \dots y_{t-L}\right\} \end{aligned} \tag{5.1}$$

5.2 Experimental design

The total dataset comprised target and input values from 01/01/1997 to 12/12/2008 (3118 trading days). Various combinations of in-sample and out-of-sample periods were used for the experiments covering this dataset, but after a combination of

trial and error and the consideration of the practical implications of implementing the trading system, an in-sample period of one year (262 trading days) used to train the networks to make predictions for the following six months (131 days) was decided on[1]. Logarithms of five and one day returns of this carry basket were used as target values along with the five exogenous variables and three lags of the carry basket returns as input variables.

Experiments were conducted by training the networks for one year, outputting predicted values for six months, rolling on six months so that the following out-of-sample started at the end of the previous in-sample and recording the out-of-sample predictions for the 21 periods that the dataset encompassed in this manner. The time-series of predictions generated using this rolling window of predictions, which encompassed 2751 trading days from 06/01/1998 to 22/07/2008, was used as the input to various simple trading rules so that the cumulative effect of asset returns as if the FX carry basket had been traded could be ascertained.

Neural Networks with one hidden layer using various activation functions, regularisation parameters and numbers of hidden neurons were investigated. The effect of pre-processing the inputs using standard normalization methods as well as Principal Component Analysis was researched. After the activation function and a pre-processing method had been decided upon, different numbers of hidden neurons and regularization parameters were used for the several ANN used at the committee stage.

SVM and RVM using radial and linear kernels, alternative pre-processing methods and parameters for C, ϵ and σ (where relevant) were investigated. After settling on a kernel and pre-processing method, different values for C, ϵ and σ were used for the SVM and RVM when used at the committee stage.

The ANN, SVM and RVM were used both in an attempt to predict actual one day and five day returns and also to predict whether the one/five day return was below various threshold values. It was found that the latter implementation of the

[1]Note that the in and out-of sample lengths are in themselves parameters and one could argue that they should therefore not have been selected from test data in the fashion described here, however the sensitivity of the performance to significant changes in this parameter was minimal and it was consequently felt that selecting the window lengths in this manner would not lead to over-fitting.

networks, i.e. for classification, was much more effective. However, the ANN, SVM and RVM performed differently from each other, depending on how negatively the threshold value was set. Three alternative values for the threshold were therefore used when the classifiers were combined at the committee stage.

Various combinations of different implementations (i.e. parameter settings, number of hidden neurons, kernels etc.) of ANN, SVM and RVM in conjunction with each other were investigated and an optimal committee comprising of the predictions of ten classifiers was decided upon. These ten predictions were fed into various simple trading rules to generate trading signals, informing a trader to what extent he should be long/short the basket on any given day. It was found that in general the committee of networks was much more effective at predicting five day returns than one day returns, and it was on this basis that the optimal configuration was used.

5.3 Results

Conservative transaction costs of 0.04%[2] of the position size per trade were used to estimate the returns that would have been realised for the optimal trading rule based on the classifier predictions over the 21 out-of-sample periods which the dataset comprised. These are shown in Table 2 alongside the benchmark of constantly holding this long-biased asset - see Section 17.4 in the appendix for an explanation of the acronyms.

[2]Trading costs for the currencies such as the ones used in the carry basket are typically one or two basis points (one hundredths of a percent), only very occasionaly increasing to three or four basis points during illiquid times.

Figure 1: FX carry basket cumulative returns for committee prediction and benchmark

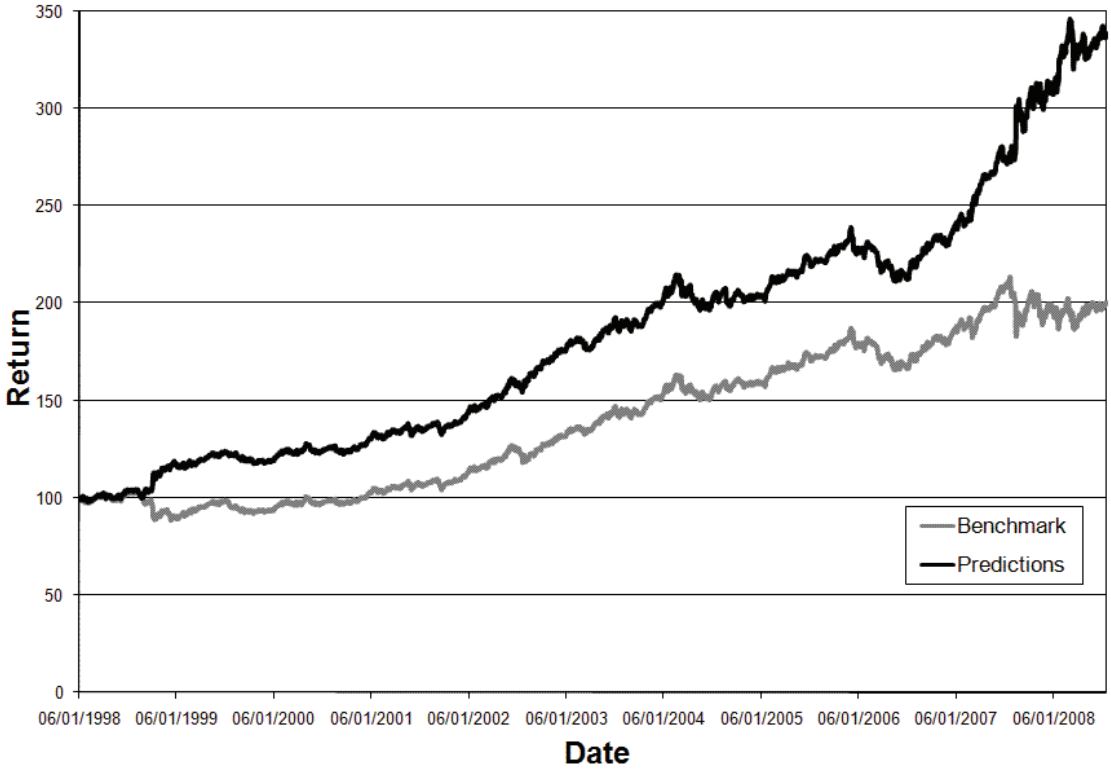


Figure 1 details the actual time series of returns using the classifier predictions alongside the benchmark and hence highlights the many occasions when negative returns could have been pre-empted and the trader would have profited by going short the basket. In this sense, the classifier predictions are only able to outperform the benchmark in periods when it falls significantly. This is most evident in the final two and a half year period on the graph.

Table 2: Comparison between benchmark and committee prediction

Time Series	Always-In Benchmark	Committee Prediction
Overall Return	201%	339%
CAGR (%)	6.88%	12.36%
SD Annualized	8.4%	8.3%
SD Loss Annualized	6.9%	5.9%
Max Draw Down	-14.2%	-11.6%
Max DD Time (days)	628	295
Sharpe Ratio	0.82	1.49
Sortino Ratio	0.99	2.08

5.4 Conclusions

Assuming conservative estimates of trading costs, over the 10.5 year (2751 trading day) rolling out-of-sample period investigated, 110% in Sortino and 80% in Sharpe relative to the 'Always In' benchmark were found. Furthermore, the extent of the maximum draw-down was reduced by 19% and the longest draw-down period was 53% shorter.

6 Multiple Kernel Learning on the limit order book

Following the success of the committee of discriminative techniques in predicting FX carry basket returns, it was felt that greater innovation on the theoretical side should be attempted in order to ascertain whether making the predictive methods richer from a machine learning perspective would improve performance.

The work in this section was presented at the Workshop on Applications of Pattern Analysis (WAPA) 2010 and is published in the Journal of Machine Learning Research (JMLR) Proceedings Series [1].

6.1 Background

A trader wishing to speculate on a currency's movement is most interested in what direction he believes that currency will move over a forecast horizon Δt so that he can take a position based on this prediction and speculate on its movement: buying the currency (going long) if he believes it will go up and selling it (going short) if he believes it will go down.

A method which deals with the problem of kernel selection is that of Multiple Kernel Learning (MKL) (e.g. [108], [109]). This technique mitigates the risk of erroneous kernel selection to some degree by taking a set of kernels and deriving a weight for each kernel such that predictions are made based on a weighted sum of several kernels.

Multiple Kernel Learning considers convex combinations of K kernels:

$$\boldsymbol{\kappa}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{t=1}^{K} d_t \kappa_t(\boldsymbol{x}_i, \boldsymbol{x}_j) \text{ where } \; d_t \geq 0, \; \sum_t d_t = 1. \tag{6.1}$$

It is often useful when looking at high dimensional datasets (for example in the world of computer vision) to summarise them by extracting features (e.g. [110]). Taking this into account, the principle of MKL can be extended by constructing a set of kernels that not only includes different kernel functions applied to the input data but also different features of the data itself. The weightings that the MKL allocates to each kernel method / feature combination highlights its usefulness in

representing the input space for the task at hand.

The majority of the research using SVM in financial prediction tasks deals with the problem of kernel selection in a purely empirical manner with little to no theoretical justification. The exceptions to this being Wang and Zhu (2008) [111], who use a two step kernel-selection/SVM procedure and Luss and D'Aspremont (2008) [112] who use Multiple Kernel Learning (MKL) to classify the impact of news for a financial prediction task. There is very scant evidence of research using MKL in financial market prediction and no evidence of work based on using MKL on order book volume data, with all previous research using features based on previous price movements (e.g. [113]).

For the purposes of exploring MKL on order book data, two MKL techniques will be investigated: Rakotomamonjy *et al.'s* (2008) SimpleMKL [114] and Hussain and Shawe-Taylor's (2009) LPBoostMKL [115] techniques.

6.2 SimpleMKL

SimpleMKL learns the kernel weightings (i.e. d_t of (6.1)) along with the α's by using semi-infinite linear programming to solve the following constrained optimisation:

$$\min_d J(d) = \left\{ \begin{array}{ll} \min\limits_{\{f\},b,\xi} & \frac{1}{2}\sum\limits_{t=1}^{K}\frac{1}{d_t}\left\|f_t\right\|^2 + C\sum\limits_{i=1}^{m}\xi_i \\ \text{s.t.} & y_i\sum\limits_{t=1}^{K} f_t(\boldsymbol{x_i}) + y_i b \geq 1 - \xi_i \\ & \xi_i \geq 0,\ i = 1,\ldots,m \end{array} \right\} \tag{6.2}$$

where:

$$f_t(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^{m}\alpha_i y_i \kappa_t(\boldsymbol{x_i}, \boldsymbol{x}) + b\right) \tag{6.3}$$

The associated dual problem is very similar to (2.14):

$$\begin{aligned} &\min_{d_t} \left\{ \max_{\alpha} -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \sum_t d_t \kappa_t(x_i, x_j) + \sum_i \alpha_i \right\} \\ &\text{s.t. } \sum_t d_t = 1, \;\; d_t \geq 0, \;\; \sum_i \alpha_i y_i = 0, \;\; C \geq \alpha_i \geq 0 \end{aligned} \tag{6.4}$$

This problem can be solved using a standard SVM optimisation solver for a fixed weighting of kernels, to generate an initial solution for $\boldsymbol{\alpha}$. After this step one can fix $\boldsymbol{\alpha}$ and use a linear programming technique to solve for the d's and hence find the weights of the kernels. This 2-step process is repeated until a convergence criterion is met.

6.3 LPBoostMKL

In general, boosting consists of constructing a convex combination of *weak* learners - defined as classifiers that misclassify less than half the time. The expectation is that a weighted combination of weak learners may be boosted to become a single strong learner, see for example [116]. This is possible if each of the weak learners are slightly correlated with the true classification but not fully correlated with each other.

The novel LPBoostMKL methodology, which is an extension of Linear Programming Boosting via Column Generation [117], treats each kernel κ_t as defining a class of weak learners and derives the weightings of the kernel through the following optimisation [115]:

$$\begin{aligned} \min \quad & \beta \\ s.t. \quad & \textstyle\sum_{i=1}^m u_i y_i h_{t(p)}(\boldsymbol{x_i}) \leq \beta, \; p = 1, \ldots, n \\ & \textstyle\sum_{i=1}^m u_i = 1 \\ & 0 \leq u_i \leq D \end{aligned}$$

where n are the number of weak learners chosen and

$$h_{t(p)}(\boldsymbol{x_i}) = \frac{1}{v_{t(p)}} \sum_{i=1}^{m} u_i y_i \kappa_{t(p)}(\boldsymbol{x_i}, \boldsymbol{x}) \tag{6.5}$$

is the weak learner constructed using the kernel $\kappa_{t(p)}$, where $t(p) \in \{1, \ldots, K\}$.

The weak learner is normalised using $v_{t(p)}$:

$$v_{t(p)} = \max_{t(p)\in\{1,\ldots,K\}} \sqrt{\sum_{i,j=1}^{m} u_i u_j y_i y_j \kappa_{t(p)}(\boldsymbol{x_i}, \boldsymbol{x_j})} > \beta. \tag{6.6}$$

When the weak learners are constructed using Equation (6.5), LPBoost solves the MKL problem [115].

6.4 Experimental design

Representing the volume at time t at each of the price levels of the order book on both sides as a vector $\boldsymbol{V}_t$ (where $\boldsymbol{V}_t \in \mathbb{R}^6_+$ for the case of three price levels on each side) a set of features was constructed:

$$\mathcal{F} = \left\{ \boldsymbol{V}_t, \frac{\boldsymbol{V}_t}{\|\boldsymbol{V}_t\|_1}, \boldsymbol{V}_t - \boldsymbol{V}_{t-1}, \frac{\boldsymbol{V}_t - \boldsymbol{V}_{t-1}}{\|\boldsymbol{V}_t - \boldsymbol{V}_{t-1}\|_1} \right\}$$

Radial Basis Function kernels have often proved useful in financial market prediction problems, e.g. [89], and for this reason a set consisting of three radial kernels with different values of σ^2 along with a linear kernel was used:

$$\mathcal{K} = \left\{ \exp\left(\frac{-\|\boldsymbol{x_i} - \boldsymbol{x_j}\|^2}{\sigma_1^2}\right), \ldots, \exp\left(\frac{-\|\boldsymbol{x_i} - \boldsymbol{x_j}\|^2}{\sigma_5^2}\right), \langle \boldsymbol{x_i}, \boldsymbol{x_j} \rangle \right\}$$

This meant that altogether there were $|\mathcal{F}| \times |\mathcal{K}| = 4 \times 6 = 24$ feature / kernel combinations constructed from the order book data, so that for example the combination $\mathcal{K}_1\mathcal{F}_1$ is the Radial Basis Function with the scale parameter σ_1^2 applied to the simplest volume feature $\boldsymbol{V}_t$ and $\mathcal{K}_3\mathcal{F}_6$ is the linear kernel applied to the volume feature $\boldsymbol{V}_t - \boldsymbol{V}_{t-1}$.

In the problem of currency prediction, any move that is predicted has to be significant enough to cross the spread in the appropriate direction if the trader is to profit from it and here we are concerned with a multiclass classification of that movement. We attempt to predict whether the trader should go long by buying the currency pair because $P^{Bid}_{t+\Delta t} > P^{Ask}_t$, go short by selling it because $P^{Ask}_{t+\Delta t} < P^{Bid}_t$ or do nothing because $P^{Bid}_{t+\Delta t} < P^{Ask}_t$ and $P^{Ask}_{t+\Delta t} > P^{Bid}_t$.

With this in mind, three SVM are trained on the data with the following labelling

criteria for each SVM:

$$\begin{aligned}
&\text{SVM 1:} P^{Bid}_{t+\Delta t} > P^{Ask}_{t} && \Rightarrow y^1_t = +1, \text{ otherwise } y^1_t = -1 \\
&\text{SVM 2:} P^{Ask}_{t+\Delta t} < P^{Bid}_{t} && \Rightarrow y^2_t = +1, \text{ otherwise } y^2_t = -1 \\
&\text{SVM 3:} P^{Bid}_{t+\Delta t} < P^{Ask}_{t}, P^{Ask}_{t+\Delta t} > P^{Bid}_{t} && \Rightarrow y^3_t = +1, \text{ otherwise } y^3_t = -1
\end{aligned}$$

In this manner, a three dimensional output vector $\boldsymbol{y}_t$ is constructed from y^1_t, y^2_t and y^3_t for each instance such that $\boldsymbol{y}_t = [\pm 1, \pm 1, \pm 1]$. Predictions are only kept for instances where exactly one of the signs in $\boldsymbol{y}_t$ is positive, i.e. when all three of the classifiers are agreeing on a direction of movement. For this subset of the predictions, a prediction is deemed correct if it correctly predicts the direction of spread-crossing movement (i.e. upwards, downwards or no movement) and incorrect if not.

Both MKL methods described above were investigated along with standard SVM based on each of the 24 kernels / feature combinations individually. Predictions for time horizons (Δt) of 5, 10, 20, 50, 100 and 200 seconds into the future were created. Training and prediction were carried out by training the three SVM on 100 instances of in-sample data, making predictions regarding the following 100 instances and then rolling forward 100 instances so that the out-of-sample data points in the previous window became the current window's in-sample set. The data consisted of 6×10^4 instances of order book updates for the EURUSD currency pair from the EBS exchange starting on 2/11/2009. EURUSD was selected as the currency pair to investigate because it is the most actively traded currency pair, comprising 27% of global turnover [29]. Consequently, the EBS exchange was selected for this analysis because it is the primary ECN for EURUSD, meaning that most of the trading for the pair takes place on it.

6.5 Results

When describing the predictive accuracy of the three different kernel methods (SimpleMKL, LPBoostMKL and the individual kernels) several factors need to be considered: how often each method was able to make a prediction as described above, how correct the predictions were overall for the whole dataset and how the predictive accuracy varied depending on the direction of movement predicted, e.g. how many predicted upward movements actually transpired, etc. In the tables

and figures that follow, for the sake of clarity only one of the 24 individual kernels is used when comparing the two MKL techniques to the individual kernels. The kernel chosen is the one with the most significant weightings from the SimpleMKL and LPBoostMKL methods (as highlighted in Figures 2 and 3), namely the radial basis function mapping with the smallest scale parameter (σ^2) on the simple volumes feature, i.e. $\mathcal{K}_1\mathcal{F}_1$.

A simple trend-following benchmark was employed for the sake of comparison. A moving average (MA) crossover technique (see for example [118]) consisting of a signal constructed from moving averages of two rolling windows, MA_{long} and MA_{short}, was used. When $MA_{long} < MA_{short}$ the signal was to go long (+1) and when $MA_{long} > MA_{short}$ the signal was to go short (-1). The window lengths chosen for the two periods were those that gave the highest predictive accuracy for the data set. This technique was chosen as a benchmark because out of the commonly used technical analysis methods toolset it is one of the simplest and when the parameters have been optimised retrospectively to suit the overall dataset (in a manner that would not be possible in reality) it represents the most competitive comparison for the techniques investigated here. In contrast to the kernel-based methods, this rule produced a continuous non-zero signal.

Table 3 shows how often each of the methods were able to make a prediction for each of the time horizons and Table 4 shows each of the methods' predictive accuracy over the entire dataset when a prediction was actually possible.

Table 3: % number of instances predictions possible

Δt	**SimpleMKL**	**LPBoostMKL**	$\mathcal{K}_1\mathcal{F}_1$	**Moving Average**
5	27	24	26	100
10	43	38	42	100
20	51	50	50	100
50	44	43	43	100
100	35	34	35	100
200	26	25	21	100

Tables 5, 6 and 7 break down the predictive accuracy of each method conditioned on whether a positive, negative or no movement (i.e. not spread-crossing) were predicted. Note that "NA" indicates that no predictions for that particular value

Table 4: % accuracy of entire dataset

Δt	**SimpleMKL**	**LPBoostMKL**	$\mathcal{K}_1\mathcal{F}_1$	**Moving Average**
5	95	95	95	4
10	90	90	89	6
20	81	81	81	10
50	66	66	66	17
100	51	51	51	24
200	45	46	46	30

of Δt were possible.

Table 5: % accuracy of positive movement predictions

Δt	**SimpleMKL**	**LPBoostMKL**	$\mathcal{K}_1\mathcal{F}_1$	**Moving Average**
5	NA	NA	NA	2
10	NA	NA	NA	5
20	12	33	6	9
50	30	32	30	17
100	27	27	27	24
200	35	35	38	30

Table 6: % accuracy of negative movement predictions

Δt	**SimpleMKL**	**LPBoostMKL**	$\mathcal{K}_1\mathcal{F}_1$	**Moving Average**
5	NA	NA	NA	2
10	NA	NA	NA	4
20	16	17	16	8
50	16	17	16	16
100	19	21	21	23
200	34	37	39	31

Figures 2 and 3 show the average weighting that SimpleMKL and LPBoostMKL assigned to each of the 24 individual feature / kernel combinations.

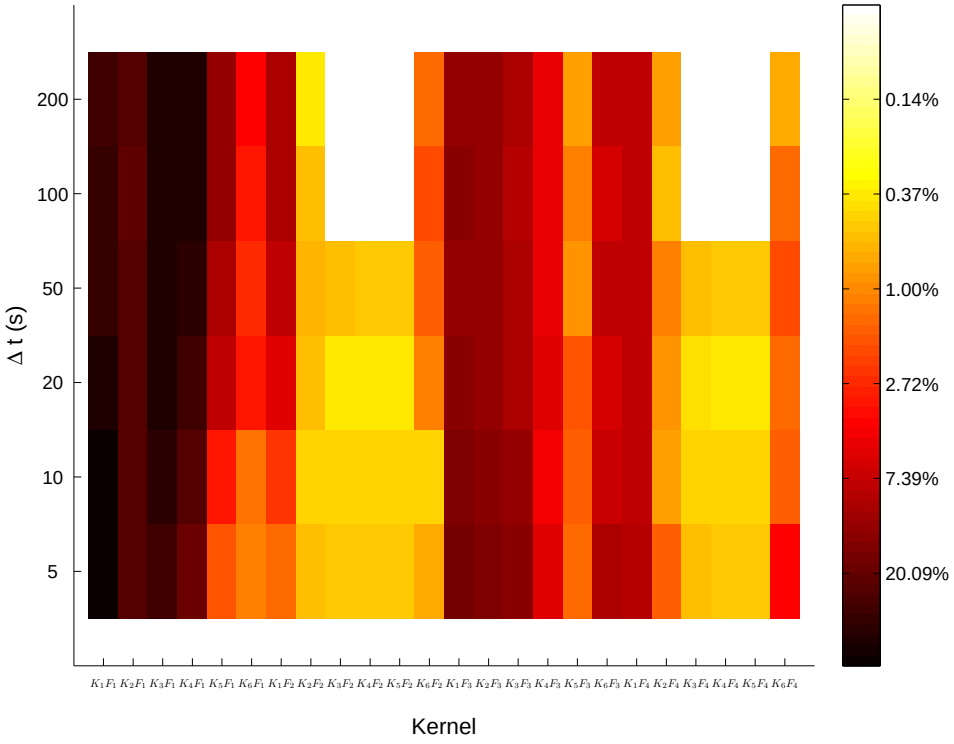
6.6 Conclusions

Table 3 indicates that both MKL methods and the individual kernel were able to make predictions between a third and half the time with the MKL methods slightly

Table 7: % accuracy of zero movement predictions

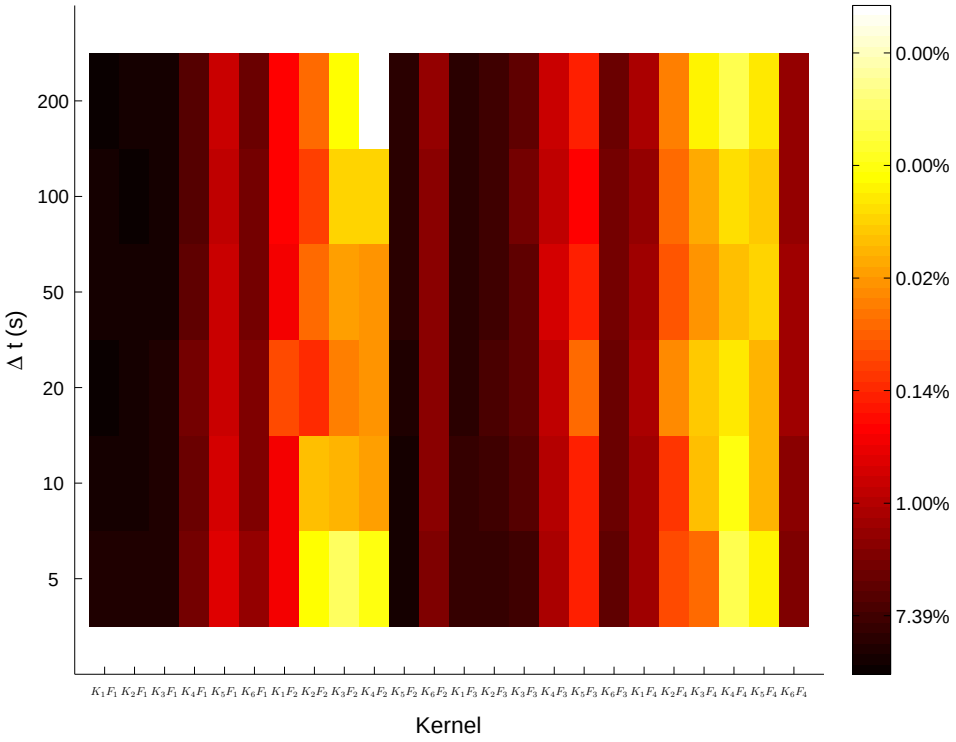
Δt	**SimpleMKL**	**LPBoostMKL**	$\mathcal{K}_1\mathcal{F}_1$	**Moving Average**
5	95	95	95	NA
10	90	90	89	NA
20	82	81	82	NA
50	70	70	70	NA
100	59	60	60	NA
200	50	50	50	NA

Figure 2: Graph of SimpleMKL kernel weightings



outperforming the individual kernels in the majority of cases. Table 4 shows that the individual kernels and MKL methods are significantly better than the MA benchmark, despite the latter having had the unfair advantage of having had its two parameters optimised on this performance metric. However, the main reason for this is because the MA technique is not able to predict zero movements. When the predictive accuracy is conditioned on the direction of the movement predicted, as shown in Tables 5, 6 and 7, one can see that although the kernel methods still outperform the trend-following technique for positive and negative movement predictions in the majority of cases, the out-performance is less significant.

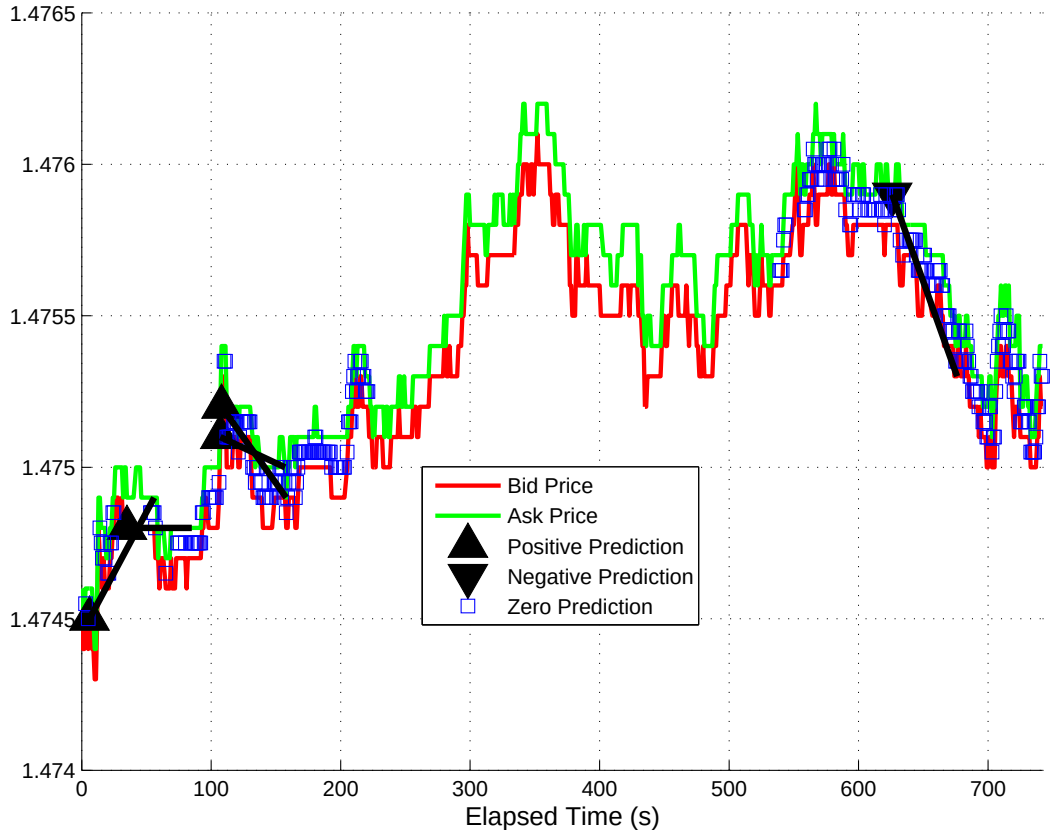
Figure 3: Graph of LPBoostMKL kernel weightings



Figures 2 and 3 show that for both MKL techniques, kernels $\mathcal{K}_1\mathcal{F}_1$, $\mathcal{K}_1\mathcal{F}_2$, $\mathcal{K}_1\mathcal{F}_3$, $\mathcal{K}_3\mathcal{F}_1$, $\mathcal{K}_3\mathcal{F}_2$ and $\mathcal{K}_3\mathcal{F}_3$ have consistently high weightings and hence were the most relevant for making predictions over the data set. These kernels are the radial basis function mapping with the three smallest scale parameters (σ^2) on the simple volumes feature and the change in volumes feature. The fact that both MKL methods selected very similar weightings for the 24 different mapping / feature combinations highlights the consistency of the techniques. Furthermore, the vertical bands of colour (or intensity) highlight the consistency of each of the kernel / feature combination's weightings across the different time horizons: in almost all cases the weighting for a particular combination is not significantly different between when being used to make a prediction for a short time horizon and a longer term one.

Figure 4 shows a 700 second snap-shot of the positive and negative predictions generated by the SimpleMKL method for $\Delta_t = 50$. The triangles denoting predictions of positive or negative movement (at the best bid or ask price respectively) have black lines linking them to the opposing (spread-crossing) price at the end of each prediction's 50 second forecast horizon. In this particular example, one can

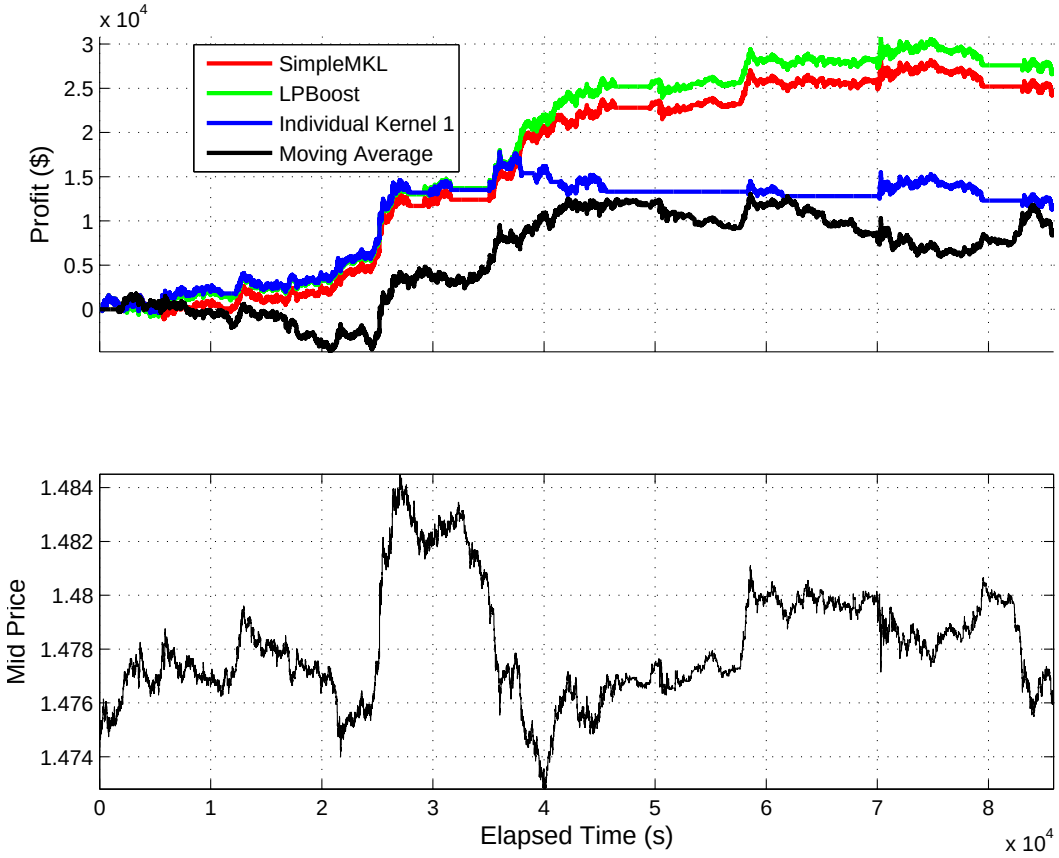
Figure 4: Example of SimpleMKL predictions for 50s horizon with resulting actual price movement



see that the SimpleMKL method commences by making four predictions for positive movement, the first of which is significantly correct, followed by an even more successful downward movement prediction. The many zero movement predictions shown do not have their resulting predictions plotted for the sake of clarity. If nothing else, this graphically depicts the fact that although these techniques may make incorrect predictions more often than correct ones, the extent of the moves that work in the technique's favour more than compensate for the losses of the incorrect ones.

The upper half of Figure 5 shows a theoretical Profit and Loss (P&L) curve for the period constructed from a trading rule based on each of the technique's signals. Amongst other things, it was assumed that one had to pay the spread each time a trade was made, that the trade size each time was $1M and that there was no slippage. It highlights the fact that similar predictive accuracies do not necessarily translate into similar P&L curves, with the two MKL techniques clearly outperforming the individual kernel and all three of the kernel methods outperforming

Figure 5: P&L curves of techniques (above) and time series of EURUSD over dataset (below)



the trend-following one. The profit that the two MKL methods realise in this example are significant both in their magnitude and also in their consistency over time, in that the periods where money is lost (draw-downs) are both infrequent and short-lived. The lower half of the figure shows the price of EURUSD over the time period that the dataset encompasses, which clearly contained no overall trends in either direction.

In terms of comparing the two MKL methods, it is worth noting that they are both solving the same optimisation problem (as expressed in (6.1)), so one would expect them to give very similar results, as they do in the majority of cases. However, one of the main differences between them is that the continuously improving LPBoost algorithm can be stopped at any point prior to convergence to produce a suboptimal classifier and in this sense one can control the accuracy vs training time trade-off for the method. This aspect of the method is of practical benefit in real-time applications where training time is an important constraint.

The results of the MKL experiments described here are significant in that no price action or features based on prices is taken into account when predicting future prices - in stark contrast to other similar research. This means that any trading rules based on this technique are likely to complement existing rules well, the majority of which look at previous price action in some manner or other. Furthermore, the out-performance of the kernel-based techniques for long time horizons over the trend-following benchmark make them a useful method for locating turning points in time series of EURUSD prices.

7 Currency forecasting using MKL with financially motivated features

Given the success of using MKL methods with fairly simple features, it was felt that augmenting the features used to embody standard concepts from the world of trading could potentially improve performance. Furthermore, because of the significant similarity between LPBoostMKL and SimpleMKL, there seemed little to be gained by continuing to compare the two techniques and research on LPBoostMKL was discontinued at this point.

The work in this section was presented at the NIPS2010 Workshop: New Directions in Multiple Kernel Learning.

7.1 Background

The following four Price-based Features are based on common price-based trading rules (which are described briefly in Section 17.5 of the appendix):

$$\mathcal{F}_1 = \left\{ EMA_t^{L_1}, \ldots, EMA_t^{L_N} \right\}$$

$$\mathcal{F}_2 = \left\{ MA_t^{L_1}, \ldots, MA_t^{L_N}, \sigma_t^{L_1}, \ldots, \sigma_t^{L_N} \right\}$$

$$\mathcal{F}_3 = \left\{ P_t, \max_t^{L_1}, \ldots, \max_t^{L_N}, \min_t^{L_1}, \ldots, \min_t^{L_N} \right\}$$

$$\mathcal{F}_4 = \left\{ \Uparrow_t^{L_1}, \ldots, \Uparrow_t^{L_N}, \Downarrow_t^{L_1}, \ldots, \Downarrow_t^{L_N} \right\}$$

where $EMA_t^{L_i}$ denotes an exponential moving average of the price P at time t with a half life L_i, $\sigma_t^{L_i}$ denotes the standard deviation of P over a period L_i, $MA_t^{L_i}$ its simple moving average over the period L_i, $\max_t^{L_i}$ and $\min_t^{L_i}$ the maximum and minimum prices over the period and $\Uparrow_t^{L_i}$ and $\Downarrow_t^{L_i}$ the number of price increases and decreases over it.

In a manner similar to Section 6.4, a further set of four volume-based features was constructed:

$$\mathcal{F}_{5\ldots8} = \left\{ \boldsymbol{V}_t, \frac{\boldsymbol{V}_t}{\|\boldsymbol{V}_t\|_1}, \boldsymbol{V}_t - \boldsymbol{V}_{t-1}, \frac{\boldsymbol{V}_t - \boldsymbol{V}_{t-1}}{\|\boldsymbol{V}_t - \boldsymbol{V}_{t-1}\|_1} \right\}$$

7.2 Experimental design

Radial Basis Function (RBF) and polynomial kernels have often been used in financial market prediction problems, e.g. [89] and [119]. Furthermore, Artificial Neural Networks (ANN) are often used in financial forecasting tasks (e.g. [80, 81, 82]) and for this reason a kernel based on Williams (1998) [120] Infinite Neural Network (INN) with a sigmoidal transfer function is also employed (see $\mathcal{K}_{11:15}$ below). A feature mapping set consisting of five of each of these kernel types with different values of the relevant hyperparameter (σ, d or Σ) along with the linear kernel is used:

$$
\begin{aligned}
\mathcal{K}_{1:5} &= \left\{ \exp\left(- \left\| \boldsymbol{x} - \boldsymbol{x}' \right\|^2 / \sigma_1^2 \right), \ldots, \exp\left(- \left\| \boldsymbol{x} - \boldsymbol{x}' \right\|^2 / \sigma_5^2 \right) \right\} \\
\mathcal{K}_{6:10} &= \left\{ \left(\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + 1 \right)^{d_1}, \ldots, \left(\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + 1 \right)^{d_5} \right\} \\
\mathcal{K}_{11:15} &= \left\{ \frac{2}{\pi} \sin^{-1} \left(\frac{2 \boldsymbol{x}^T \Sigma_1 \boldsymbol{x}'}{\sqrt{(1 + 2 \boldsymbol{x}^T \Sigma_1 \boldsymbol{x})(1 + 2 \boldsymbol{x}'^T \Sigma_1 \boldsymbol{x}')}} \right), \ldots, \frac{2}{\pi} \sin^{-1} \left(\frac{2 \boldsymbol{x}^T \Sigma_5 \boldsymbol{x}'}{\sqrt{(1 + 2 \boldsymbol{x}^T \Sigma_5 \boldsymbol{x})(1 + 2 \boldsymbol{x}'^T \Sigma_5 \boldsymbol{x}')}} \right) \right\} \\
\mathcal{K}_{16} &= \{ \langle \boldsymbol{x}, \boldsymbol{x}' \rangle \}
\end{aligned}
$$

This means that altogether there are $|\mathcal{F}| \times |\mathcal{K}| = 8 \times 16 = 128$ feature / kernel combinations. We will adopt notation so that for example the combination $\mathcal{F}_1 \mathcal{K}_1$ is the moving average crossover feature with a RBF using the scale parameter σ_1^2.

The experimental procedure as described in Section 6.4 was adopted here, however instead of comparing SimpleMKL and LPBoostMKL, only SimpleMKL was investigated along with standard SVM based on each of the 128 kernels / feature combinations individually.

When comparing the predictive accuracy of the kernel methods when used individually to their combination in MKL, one needs to consider both how often each method was able to make a prediction as described above and how correct the predictions were overall for the whole dataset. In the tables and figures that follow, for the sake of clarity only three of the 128 individual kernels are used when comparing SimpleMKL to the individual kernels. 10-fold cross-validation was used to select the three kernels with the highest predictive accuracy for the dataset, namely $\mathcal{F}_8 \mathcal{K}_{16}$, $\mathcal{F}_1 \mathcal{K}_1$ and $\mathcal{F}_1 \mathcal{K}_3$.

Table 8, which shows how often each of the methods was able to make a prediction for each of the time horizons, indicates that SimpleMKL was very similar in the frequency with which it was able to make predictions as the three individual kernel / feature combinations highlighted. Table 9 shows each of the methods'

Table 8: Percentage of time predictions possible

Δt	**SimpleMKL**	$\mathcal{F}_8\mathcal{K}_{16}$	$\mathcal{F}_1\mathcal{K}_1$	$\mathcal{F}_1\mathcal{K}_3$
5	26.1	24.7	26.1	24.7
10	41.1	40.4	39.8	37.7
20	50.2	49.1	48.1	45.0
50	46.3	44.1	44.8	45.5
100	32.8	33.5	34.6	35.3
200	27.0	24.9	26.6	27.4

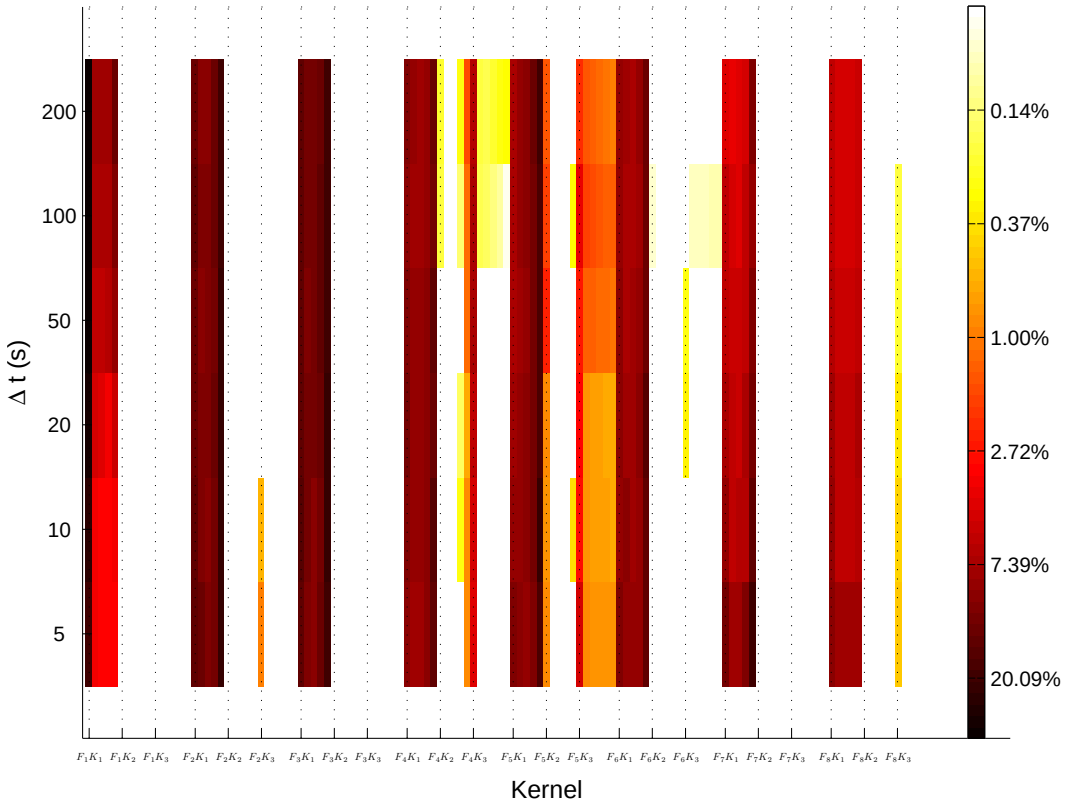
Table 9: Percentage accuracy of predictions

Δt	**SimpleMKL**	$\mathcal{F}_8\mathcal{K}_{16}$	$\mathcal{F}_1\mathcal{K}_1$	$\mathcal{F}_1\mathcal{K}_3$
5	94.7	94.7	93.0	92.8
10	89.9	89.6	88.4	84.6
20	81.7	81.3	79.5	72.3
50	67.1	65.4	65.5	61.1
100	61.1	51.1	60.7	59.9
200	58.9	45.0	58.8	61.3

predictive accuracy over the entire dataset when a prediction was actually possible. The results indicate that SimpleMKL has higher predictive accuracy than the most effective individual kernels for all time horizons under 200 seconds and is only marginally less effective than $\mathcal{F}_1\mathcal{K}_3$ for the 200 second forecast horizon.

P-values for the null hypothesis that the results reported could have occurred by chance were calculated (the methodology for doing this is explained in Section 17.6 in the appendix). It was found that for both SimpleMKL and the individual kernels highlighted for all forecast horizons, the null hypothesis could be rejected for a significance level of $< 10^{-5}$.

Figure 6: MKL Kernel weightings



As reflected in Figure 6, the kernel / feature combinations $\mathcal{F}_1\mathcal{K}_1$, $\mathcal{F}_2\mathcal{K}_5$ and $\mathcal{F}_3\mathcal{K}_5$ are consistently awarded the highest weightings by SimpleMKL and hence are the most relevant for making predictions over the data set. These kernels are the RBF mapping with the smallest scale parameter on the exponential moving average crossover feature, the RBF mapping with the largest scale parameter on the price standard deviation / moving average feature and the RBF mapping with the largest scale parameter again on the minimums / maximums feature.

The vertical banding of colour (or intensity) highlights the consistency of each of the kernel / feature combination's weightings across the different time horizons: in almost all cases the weighting for a particular combination is not significantly different between when being used to make a prediction for a short time horizon and a longer term one. One can also see from Figure 6 that although all eight of the features have weightings assigned to them, in most cases this is only in conjunction with the RBF kernels - the polynomial and infinite neural network based mappings being assigned weightings by MKL for only the fourth and fifth features.

The most successful individual kernels as selected by cross-validation are awarded very low weights by SimpleMKL. This reflects a common feature of trading rules where individual signals can drastically change their significance in terms of performance when used in combination. Furthermore, the outperformance of SimpleMKL to the individual kernels highlighted indicates that MKL is an effective method for combining a set of price and volume based features in order to correctly forecast the direction of price movements in a manner similar to a trading rule.

8 Fisher kernels

It was then decided to return to the domain of generative machine learning methods in an attempt to combine this class of models with the methods successfully used from the discriminative toolset.

The Fisher kernel represents a method for incorporating generative probability models into discriminative classifiers such as SVM. It also facilitates the inclusion of some canonical market micro-structural models, all of which are generative by nature, into the discriminative machine learning domain. It is this incorporation of traditional, often empirically based, market microstructural (MM) models into the machine learning framework that represents the main contribution of this work.

When one adapts the parameters of a model to incorporate a new data point so that the model's likelihood $\mathcal{L}$ will increase, a common approach is to adjust each parameter θ_i by some function of $d\mathcal{L}/d\theta_i$. The Fisher kernel [121] incorporates this principle by creating a kernel composed of values of $d\mathcal{L}/d\theta_i$ for each of the model's parameters and therefore comparing data instances by the way they stretch the model's parameters.

Defining the log likelihood of a data item x with respect to a model for a given setting of the parameters $\boldsymbol{\theta}$ to be $\log \mathcal{L}_{\boldsymbol{\theta}}(x)$, the Fisher score of x is the vector gradient of $\log \mathcal{L}_{\boldsymbol{\theta}}(x)$:

$$\mathbf{g}(\boldsymbol{\theta}, x) = \left(\frac{\partial \log \mathcal{L}_{\theta_i}(x)}{\partial \theta_i} \right)_{i=1}^{N} \tag{8.1}$$

The practical Fisher kernel is then defined as:

$$\kappa(x, z) = \mathbf{g}(\boldsymbol{\theta}, x)' \mathbf{g}(\boldsymbol{\theta}, z) \tag{8.2}$$

From the literature review, the most significant and widely adopted MM models are based around three main families: Autoregressive Conditional Duration models, Poisson processes and Wiener processes. I will briefly describe the background to each of these classes of models and how they can be incorporated into Fisher kernels.

8.1 Autoregressive Conditional Duration model

The literature on time deformation in financial markets suggests that sometimes time, as observed through the rate of a particular financial transaction occurring, flows very rapidly while in other periods it moves slowly. The market microstructure literature, for example [122], shows that one should expect stochastic clustering in the rate of price changes for a financial asset. An explanation for this is that ill-informed traders trade randomly according to a stochastic process such as a Poisson process, while informed traders enter the market only after observing a private, potentially noisy signal. The agents providing the prices (market makers) will slowly learn of the private information by watching order flow and adjust their prices accordingly. Informed traders will seek to trade as long as their information has value. Hence one should see clustering of trading following an information event because of the increased numbers of informed traders.

Engle and Russel's (1998) Autoregressive Conditional Duration (ACD) model [123] captures this stochasticly clustering arrival rate by expressing the duration of a price (how long a financial asset's price remains constant) as a function of previous durations. The model is used commonly throughout the market microstructure literature, e.g. to measure the duration of financial asset prices [124]. It is defined as follows:

$$h_t = w + \sum_{i=1}^{L} q_i x_{t-i} + \sum_{i=1}^{L} p_i h_{t-i} \tag{8.3}$$

$$x_t = h_t \epsilon_t, \quad \epsilon_t \sim Exp(\lambda) \tag{8.4}$$

where x_t is the duration of the price at time t, h_t is its expected duration, L is the lag of the autoregressions and w, $\boldsymbol{p}$, $\boldsymbol{q}$ and λ are constants.

Simplifying (8.3) to be an order 1 autoregressive process, the likelihood of the model can be expressed:

$$\mathcal{L} = \lambda \exp\left[-\lambda x_t (w + q x_{t-1} + p h_{t-1})^{-1}\right] \tag{8.5}$$

Differentiating $\mathcal{L}$ wrt each of the parameters:

$$\frac{\partial \mathcal{L}}{\partial w} = \lambda^2 x_t (w + q x_{t-1} + p h_{t-1})^{-2} \exp\left[-\lambda x_t (w + q x_{t-1} + p h_{t-1})^{-1}\right] \tag{8.6}$$

$$\frac{\partial \mathcal{L}}{\partial q} = \lambda^2 x_t x_{t-1} (w + q x_{t-1} + p h_{t-1})^{-2} \exp\left[-\lambda x_t (w + q x_{t-1} + p h_{t-1})^{-1}\right] \tag{8.7}$$

$$\frac{\partial \mathcal{L}}{\partial p} = \lambda^2 x_t h_{t-1} (w + q x_{t-1} + p h_{t-1})^{-2} \exp\left[-\lambda x_t (w + q x_{t-1} + p h_{t-1})^{-1}\right] \tag{8.8}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = (1 - \lambda x_t (w + q x_{t-1} + p h_{t-1})^{-1}) \exp\left[-\lambda x_t (w + q x_{t-1} + p h_{t-1})^{-1}\right] \tag{8.9}$$

The chain rule can then be used to express these differentials as functions of the log likelihood ($\mathcal{LL}$):

$$\frac{d\left(\ln\left[\mathrm{f}(x)\right]\right)}{dx} = \frac{\mathrm{f}'(x)}{\mathrm{f}(x)} \Rightarrow$$

$$\frac{\partial \mathcal{LL}}{\partial w} = \lambda x_t (w + q x_{t-1} + p h_{t-1})^{-2} \tag{8.10}$$

$$\frac{\partial \mathcal{LL}}{\partial q} = \lambda x_t x_{t-1} (w + q x_{t-1} + p h_{t-1})^{-2} \tag{8.11}$$

$$\frac{\partial \mathcal{LL}}{\partial p} = \lambda x_t h_{t-1} (w + q x_{t-1} + p h_{t-1})^{-2} \tag{8.12}$$

$$\frac{\partial \mathcal{LL}}{\partial \lambda} = \lambda^{-1} - x_t (w + q x_{t-1} + p h_{t-1})^{-1} \tag{8.13}$$

A search algorithm, such as the simplex method [125], can be used to find estimates for p, q and w based on the observed durations $\boldsymbol{x}_{1:T}$ and then these estimates can be used in (8.10) - (8.13) in order to derive the Fisher score of a new observation for each of the four parameters. This will be done for price durations on the front bid x_{Bid} and x_{Ask} sides so that an 8 dimensional Fisher score vector can be calculated for each data instance:

$$\mathbf{g}_t^{ACD} = \left\{ \frac{\partial \mathcal{LL}_t}{\partial w_{Bid}}, \frac{\partial \mathcal{LL}_t}{\partial q_{Bid}}, \frac{\partial \mathcal{LL}_t}{\partial p_{Bid}}, \frac{\partial \mathcal{LL}_t}{\partial \lambda_{Bid}}, \frac{\partial \mathcal{LL}_t}{\partial w_{Ask}}, \frac{\partial \mathcal{LL}_t}{\partial q_{Ask}}, \frac{\partial \mathcal{LL}_t}{\partial p_{Ask}}, \frac{\partial \mathcal{LL}_t}{\partial \lambda_{Ask}} \right\} \tag{8.14}$$

8.2 Poisson processes

Poisson processes permeate the market microstructure literature in their descriptions of limit order arrival rates, volume changes and order cancellations - see for example [126]. However, their modelling of trade arrival rates are subject to criticism and it is suggested that renewal processes might be better placed to do this - see for example [127], [128], [129] and [130].

If we denote the volume at each depth i of the order book ($i \in [1 \dots 2D]$ assuming D levels on the bid side and D on the ask) at time t as V_t^i, we can use ΔV_i to represent the rate of change of volume at a depth over a given time interval τ:

$$\Delta V_i = \frac{\left| V_{t+\tau}^i - V_t^i \right|}{\tau} \tag{8.15}$$

We can model ΔV_i using a Poisson process, i.e. $\Delta V_i \sim Poi(\lambda_i)$:

$$P(\Delta V_i = x) = \frac{e^{-\lambda_i \tau} (\lambda_i \tau)^x}{x!} \tag{8.16}$$

Setting the time interval τ to 1, the log likelihood of a rate x_i observed at depth i for a model parameterised by λ_i can be expressed:

$$\begin{aligned} \mathcal{LL} &= \log \left(\frac{e^{-\lambda_i} (\lambda_i)^{x_i}}{x_i!} \right) \\ &= -\lambda_i + x_i \log(\lambda_i) - \log(x_i!) \end{aligned} \tag{8.17}$$

Differentiating (8.17) with respect to λ_i we can derive the Fisher score:

$$\frac{\partial \mathcal{LL}}{\partial \lambda_i} = -1 + \frac{x_i}{\lambda_i} \tag{8.18}$$

The parameter λ_i needs to be estimated at each depth i. The Maximum Likelihood (ML) estimate of λ_i can be calculated by adjusting (8.17) to take into account a set of N observations:

$$\begin{aligned} \mathcal{LL}_N &= \sum_{j=1}^{N} \log \left(\frac{e^{-\lambda_i} (\lambda_i)^{x_i^j}}{x_i^j!} \right) \\ &= -N\lambda_i + \left(\sum_{j=1}^{N} x_i^j \right) \log(\lambda_i) - \sum_{j=1}^{N} \log(x_i^j!) \end{aligned} \tag{8.19}$$

Differentiating (8.19) wrt λ_i and setting this to zero yields our ML estimate of λ_i:

$$\frac{\partial \mathcal{LL}_N}{\partial \lambda_i} = -N + \left(\sum_{j=1}^{N} x_i^j \right) \frac{1}{\lambda_i} = 0$$

$$\Rightarrow \hat{\lambda}_i = \frac{1}{N} \sum_{j=1}^{N} x_i^j \tag{8.20}$$

$\hat{\lambda}_i$ is substituted into (8.18) and a $2D$ dimensional Fisher score vector created for the D levels on each side:

$$\mathbf{g}_t^{Poiss} = \left\{ \frac{\partial \mathcal{LL}_t}{\partial \lambda_1}, \ldots, \frac{\partial \mathcal{LL}_t}{\partial \lambda_{2D}} \right\} \tag{8.21}$$

8.3 Wiener process barrier model

Lancaster's (1992) Wiener process barrier model [131] assumes that the price evolution of an asset follows a Wiener process:

$$dp_t = \mu dt + \sigma dz \tag{8.22}$$

where the price of the asset p_t follows a random walk with drift μ and variance σ^2.

This means that the price movement of an asset over a time period t will be distributed[3]:

$$p_t - p_{t-1} = \Delta p_t \sim N(\mu t, \sigma^2 t) \tag{8.23}$$

and that the ML estimates of μ and σ can be ascertained from a sequence of N such price movements:

$$\hat{\mu} = \frac{1}{N} \sum_{t=1}^{N} \Delta p_t \tag{8.24}$$

$$\hat{\sigma^2} = \frac{1}{N} \sum_{t=1}^{N} (\Delta p_t - \hat{\mu})^2 \tag{8.25}$$

Modelling the price with such a process means that the likelihood of a limit order priced at a distance α from the current mid price surviving at least t time units

[3]Note that the price movements are normally distributed as opposed to log-normally as one might expect over longer time-scales than the ones investigated here.

without being hit is:

$$\mathcal{L} = \Phi\left(\frac{\alpha - \mu t}{\sigma\sqrt{t}}\right) - \exp\left[\frac{2\mu\alpha}{\sigma^2}\right]\Phi\left(\frac{-\alpha - \mu t}{\sigma\sqrt{t}}\right) \tag{8.26}$$

where Φ represents the standardised cumulative Gaussian distribution function.

Using the substitutions $x = \frac{\alpha - \mu t}{\sigma\sqrt{t}}$, $y = \frac{-\alpha - \mu t}{\sigma\sqrt{t}}$ and $z = \exp\left[\frac{2\mu\alpha}{\sigma^2}\right]$ so that:

$$\mathcal{L} = \Phi(x) - z\Phi(y) \tag{8.27}$$

the derivatives of $\mathcal{L}$ wrt μ and σ can be calculated:

$$\begin{aligned}
\frac{\partial\mathcal{L}}{\partial\mu} &= \frac{\partial x}{\partial\mu}\frac{\partial(\Phi(x))}{\partial x} - \frac{\partial z}{\partial\mu}\Phi(y) - z\frac{\partial y}{\partial\mu}\frac{\partial(\Phi(y))}{\partial y} \\
&= -\frac{\sqrt{t}}{\sigma}\phi(x) - \frac{2\alpha}{\sigma^2}z\Phi(y) + z\frac{\sqrt{t}}{\sigma}\phi(y) \qquad (8.28) \\
\frac{\partial\mathcal{L}}{\partial\sigma} &= \frac{\partial x}{\partial\sigma}\frac{\partial(\Phi(x))}{\partial x} - \frac{\partial z}{\partial\sigma}\Phi(y) - z\frac{\partial y}{\partial\sigma}\frac{\partial(\Phi(y))}{\partial y} \\
&= -\left(\frac{\alpha - \mu t}{\sigma^2\sqrt{t}}\right)\phi(x) + \frac{4\mu\alpha}{\sigma^3}z\Phi(y) + z\left(\frac{-\alpha - \mu t}{\sigma^2\sqrt{t}}\right)\phi(y) \\
&= -\frac{x}{\sigma}\phi(x) + \frac{4\mu\alpha}{\sigma^3}z\Phi(y) + z\frac{y}{\sigma}\phi(y) \qquad (8.29)
\end{aligned}$$

where ϕ represents the standardised Gaussian distribution function.

The chain rule can then be used to express these differentials as functions of the log likelihood:

$$\frac{\partial\mathcal{L}\mathcal{L}}{\partial\mu} = \frac{-\frac{\sqrt{t}}{\sigma}\phi(x) - \frac{2\alpha}{\sigma^2}z\Phi(y) + z\frac{\sqrt{t}}{\sigma}\phi(y)}{\Phi(x) - z\Phi(y)} \tag{8.30}$$

$$\frac{\partial\mathcal{L}\mathcal{L}}{\partial\sigma} = -\frac{-\frac{x}{\sigma}\phi(x) + \frac{4\mu\alpha}{\sigma^3}z\Phi(y) + z\frac{y}{\sigma}\phi(y)}{\Phi(x) - z\Phi(y)} \tag{8.31}$$

$\hat{\mu}$ from (8.24) and $\hat{\sigma^2}$ from (8.25) can be substituted into (8.30) and (8.31) in order to derive the 4 dimensional Fisher score vector for a new set of observations:

$$\mathbf{g}_t^{Wiener} = \left\{\frac{\partial\mathcal{L}\mathcal{L}_t}{\partial\mu_{Bid}}, \frac{\partial\mathcal{L}\mathcal{L}_t}{\partial\sigma_{Bid}}, \frac{\partial\mathcal{L}\mathcal{L}_t}{\partial\mu_{Ask}}, \frac{\partial\mathcal{L}\mathcal{L}_t}{\partial\sigma_{Ask}}\right\} \tag{8.32}$$

9 Kernel parameterisation learning

9.1 Hyperparameter learning

With the aim of improving MKL's predictive performance it was decided to consider a major innovation from a machine learning perspective. Taking the MKL principle of learning the optimal weighting of a set of kernels further, one can allow aspects of the kernels themselves, for example the hyperparameter values used in the kernel mappings, to be learnt from the data itself.

Instead of selecting a range of hyperparameter values for each kernel mapping, for example the different σ^2 in the Radial Basis Function mappings as detailed in (7.2), it is possible to learn the optimal hyperparameter value for a mapping from the data itself. Furthermore, parameterisations of the features themselves can be learnt: replacing the maximum likelihood estimates used in the Fisher kernels of Section 8 with parameters that result from an optimisation.

Referring back to Equation 6.4 in Section 6.2, the optimisation that SimpleMKL carries out aims to find the set of kernel weightings d_t which minimises the following quantity:

$$\beta = \max_{\alpha} \left\{ -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \sum_t d_t \kappa_t(x_i, x_j) + \sum_i \alpha_i \right\}$$

Once d_t and α have been selected, this is equivalent to maximising the quantity:

$$\sum_{i,j} \alpha_i \alpha_j y_i y_j \sum_t d_t \kappa_t(x_i, x_j) \equiv \boldsymbol{\alpha}^T \boldsymbol{Y} \mathcal{K}(\boldsymbol{\theta}) \boldsymbol{Y} \boldsymbol{\alpha}$$

This means that for a given kernel $\mathcal{K}$ parameterised by a hyperparameter / Fisher parameter set $\boldsymbol{\theta}$, the optimum value of $\boldsymbol{\theta}$ is:

$$\begin{aligned} &\max_{\theta} \boldsymbol{\alpha}^T \boldsymbol{Y} \mathcal{K}(\boldsymbol{\theta}) \boldsymbol{Y} \boldsymbol{\alpha} \\ =&\max_{\boldsymbol{\theta}} \mathcal{C}(\boldsymbol{\theta}) \end{aligned} \tag{9.1}$$

where $\boldsymbol{\alpha}$ is ascertained from the support vectors of an SVM trained with the target values in $\boldsymbol{Y}$. The process that determines the optimal hyperparameter setting for each kernel mapping is an iterative one:

1. Choose an initial setting $\boldsymbol{\theta} = \boldsymbol{\theta}_0$ for each of the $|\mathcal{F}| \times 3 = 12 \times 3 = 36$ feature / mapping combinations. Create a set $\mathcal{S}$ composed of these 36 kernels.

2. Train MKL on $\mathcal{S}$ and record the resulting $\boldsymbol{\alpha}$, kernel weightings $\mathbf{d}$ and β. Remove any kernels from $\mathcal{S}$ which receive a zero weighting from MKL.

3. Derive new $\boldsymbol{\theta}$ for each of the 36 kernels through the optimisation in (9.1), using $\boldsymbol{\alpha}$ from previous step. Record objective function value c for each optimised kernel.

4. Add a kernel to $\mathcal{S}$ if $c > \beta$ for that kernel.

5. Repeat steps (2) to (4) until $c \leq \beta + \epsilon$ (where ϵ is a small number).

6. The final $\mathcal{S}$, $\boldsymbol{\alpha}$ and $\mathbf{d}$ are used for future predictions.

The effect of each parameter in $\boldsymbol{\theta}$ cannot be considered independently of the other parameters on the cost function. This means that more complex optimisations than simple linear search need to be used in order to derive the optimum $\boldsymbol{\theta}$. Two different optimisation methods will be investigated:

1. Sorting the individual components of $\boldsymbol{\theta}$ by the effect they have on $\mathcal{C}(\boldsymbol{\theta})$ for an in-sample data set and then optimising each value in turn using a simple linear search method, starting with the parameter that $\mathcal{C}(\boldsymbol{\theta})$ is most sensitive to. This will be termed the *Serial Optimisation* method.

2. Calculating derivatives of $\mathcal{C}(\boldsymbol{\theta})$ with respect to each of the components in $\boldsymbol{\theta}$, namely $\nabla\mathcal{C}(\boldsymbol{\theta})$, and using a gradient search method. This will be termed the *Parallel Optimisation* method.

Elaborating on point 2.:

$$\begin{aligned}\nabla\mathcal{C}(\boldsymbol{\theta}) &= \frac{\partial}{\partial\boldsymbol{\theta}}\left(\boldsymbol{\alpha}^T\boldsymbol{Y}\mathcal{K}(\boldsymbol{\theta})\boldsymbol{Y}\boldsymbol{\alpha}\right) \\ &= \boldsymbol{\alpha}^T\boldsymbol{Y}\frac{\partial\mathcal{K}(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}\boldsymbol{Y}\boldsymbol{\alpha} \qquad (9.2)\end{aligned}$$

This means we need to calculate the elements in the matrix $\dfrac{\partial\mathcal{K}(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}$ so that for each Fisher Kernel, the second order derivatives of every parameter with respect to both itself and the other parameters, i.e. $\dfrac{\partial^2\mathcal{L}\mathcal{L}}{\partial\theta_x\partial\theta_y}$ $\forall_{x,y} = 1\ldots D$, need to be calculated.

Clarification point

In order to avoid confusion further on in this document, it is important to make the distinction between what I will refer to as *learning the kernel* (or *kernel learning*) and *MKL*. The former refers to the process described above in Section 9.1, whereby kernel mapping hyperparameters and in the case of the Fisher kernels, the kernels themselves, are learnt. This process includes *MKL* as part of the learning process.

When I refer to *MKL* on its own, I mean just that - the Multiple Kernel Learning process - where kernels in a set are allocated weightings and combined but with no other learning taking place.

9.2 Kernel derivatives

A linear feature mapping on a Fisher feature parameterised by a vector $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_D\}$ takes the general form:

$$\begin{aligned}\mathcal{K}_{i,j} &= \left[\frac{\partial \mathcal{LL}_i}{\partial \theta_1}, \ldots, \frac{\partial \mathcal{LL}_i}{\partial \theta_D}\right]\left[\frac{\partial \mathcal{LL}_j}{\partial \theta_1}, \ldots, \frac{\partial \mathcal{LL}_j}{\partial \theta_D}\right]^T \\ &= \frac{\partial \mathcal{LL}_i}{\partial \theta_1}\frac{\partial \mathcal{LL}_j}{\partial \theta_1} + \ldots, + \frac{\partial \mathcal{LL}_i}{\partial \theta_D}\frac{\partial \mathcal{LL}_j}{\partial \theta_D} \\ &= \sum_{k=1}^{D} \frac{\partial \mathcal{LL}_i}{\partial \theta_k}\frac{\partial \mathcal{LL}_j}{\partial \theta_k} \end{aligned} \tag{9.3}$$

9.2.1 RBF mapping

The RBF mapping on a Fisher feature takes the general form:

$$\mathcal{K}_{i,j} = \exp\left[\frac{1}{\sigma^2}\sum_{k=1}^{D}\frac{\partial \mathcal{LL}_i}{\partial \theta_k}\frac{\partial \mathcal{LL}_j}{\partial \theta_k} - \frac{1}{2\sigma^2}\sum_{k=1}^{D}\left(\frac{\partial \mathcal{LL}_i}{\partial \theta_k}\right)^2 - \frac{1}{2\sigma^2}\sum_{k=1}^{D}\left(\frac{\partial \mathcal{LL}_j}{\partial \theta_k}\right)^2\right] \tag{9.4}$$

(9.4) can be differentiated with respect to $\boldsymbol{\theta}$:

$$\begin{aligned}\frac{\partial\left(\mathcal{K}_{i,j}\right)}{\partial \boldsymbol{\theta}} &= \frac{\partial\left(\exp\left[f(\boldsymbol{\theta})\right]\right)}{\partial \boldsymbol{\theta}} \\ &= \frac{\partial\left[f(\boldsymbol{\theta})\right]}{\partial \boldsymbol{\theta}}\exp\left[f(\boldsymbol{\theta})\right] \end{aligned} \tag{9.5}$$

where:

$$f(\boldsymbol{\theta}) = \frac{1}{\sigma^2}\left[\sum_{k=1}^{D}\frac{\partial \mathcal{LL}_i}{\partial \theta_k}\frac{\partial \mathcal{LL}_j}{\partial \theta_k} - \frac{1}{2}\sum_{k=1}^{D}\left(\frac{\partial \mathcal{LL}_i}{\partial \theta_k}\right)^2 - \frac{1}{2}\sum_{k=1}^{D}\left(\frac{\partial \mathcal{LL}_j}{\partial \theta_k}\right)^2\right] \tag{9.6}$$

and the r^{th} row of the vector $\dfrac{\partial\left[f(\boldsymbol{\theta})\right]}{\partial \boldsymbol{\theta}}$ is:

$$\frac{\partial\left[f(\boldsymbol{\theta})\right]}{\partial \theta_r} = \frac{1}{\sigma^2}\sum_{k=1}^{D}\left[\frac{\partial^2 \mathcal{LL}_i}{\partial \theta_k \partial \theta_r}\frac{\partial \mathcal{LL}_j}{\partial \theta_k} + \frac{\partial^2 \mathcal{LL}_j}{\partial \theta_k \partial \theta_r}\frac{\partial \mathcal{LL}_i}{\partial \theta_k} - \frac{\partial^2 \mathcal{LL}_i}{\partial \theta_k \partial \theta_r}\frac{\partial \mathcal{LL}_i}{\partial \theta_k} - \frac{\partial^2 \mathcal{LL}_j}{\partial \theta_k \partial \theta_r}\frac{\partial \mathcal{LL}_j}{\partial \theta_k}\right]$$

9.2.2 Polynomial mapping

The polynomial mapping on a Fisher feature takes the general form:

$$\mathcal{K}_{i,j} = \left(\sum_{k=1}^{D} \frac{\partial \mathcal{LL}_i}{\partial \theta_k} \frac{\partial \mathcal{LL}_j}{\partial \theta_k} + 1 \right)^{\alpha} \tag{9.7}$$

(9.7) can be differentiated with respect to $\boldsymbol{\theta}$:

$$\frac{\partial \left(\mathcal{K}_{i,j} \right)}{\partial \boldsymbol{\theta}} = \alpha \left(f(\boldsymbol{\theta}) + 1 \right)^{\alpha - 1} \frac{\partial \left[f(\boldsymbol{\theta}) \right]}{\partial \boldsymbol{\theta}} \tag{9.8}$$

where:

$$f(\boldsymbol{\theta}) = \sum_{k=1}^{D} \frac{\partial \mathcal{LL}_i}{\partial \theta_k} \frac{\partial \mathcal{LL}_j}{\partial \theta_k} \tag{9.9}$$

and the r^{th} row of the vector $\dfrac{\partial \left[f(\boldsymbol{\theta}) \right]}{\partial \boldsymbol{\theta}}$ is:

$$\frac{\partial \left[f(\boldsymbol{\theta}) \right]}{\partial \theta_r} = \sum_{k=1}^{D} \left[\frac{\partial^2 \mathcal{LL}_i}{\partial \theta_k \partial \theta_r} \frac{\partial \mathcal{LL}_j}{\partial \theta_k} + \frac{\partial^2 \mathcal{LL}_j}{\partial \theta_k \partial \theta_r} \frac{\partial \mathcal{LL}_i}{\partial \theta_k} \right] \tag{9.10}$$

9.2.3 Infinite Neural Network mapping

The infinite neural network mapping on a Fisher feature takes the general form:

$$\mathcal{K}_{i,j} = \frac{2}{\pi} \sin^{-1} \left(\frac{2\alpha \sum_{k=1}^{D} \frac{\partial \mathcal{LL}_i}{\partial \theta_k} \frac{\partial \mathcal{LL}_j}{\partial \theta_k}}{\sqrt{\left(1 + 2\alpha \sum_{k=1}^{D} \left(\frac{\partial \mathcal{LL}_i}{\partial \theta_k} \right)^2 \right) \left(1 + 2\alpha \sum_{k=1}^{D} \left(\frac{\partial \mathcal{LL}_j}{\partial \theta_k} \right)^2 \right)}} \right) \tag{9.11}$$

(9.11) can be differentiated with respect to $\boldsymbol{\theta}$:

$$\frac{\partial \left(\mathcal{K}_{i,j} \right)}{\partial \boldsymbol{\theta}} = \frac{2}{\pi \sqrt{1 - f(\boldsymbol{\theta})^2}} \frac{\partial \left[f(\boldsymbol{\theta}) \right]}{\partial \boldsymbol{\theta}} \tag{9.12}$$

where:

$$f(\boldsymbol{\theta}) = \frac{2\alpha \sum_{k=1}^{D} \frac{\partial \mathcal{LL}_i}{\partial \theta_k} \frac{\partial \mathcal{LL}_j}{\partial \theta_k}}{\sqrt{\left(1 + 2\alpha \sum_{k=1}^{D} \left(\frac{\partial \mathcal{LL}_i}{\partial \theta_k}\right)^2\right)\left(1 + 2\alpha \sum_{k=1}^{D} \left(\frac{\partial \mathcal{LL}_j}{\partial \theta_k}\right)^2\right)}} \tag{9.13}$$

and the r^{th} row of the vector $\frac{\partial [f(\boldsymbol{\theta})]}{\partial \boldsymbol{\theta}}$ is:

$$\frac{\partial [f(\boldsymbol{\theta})]}{\partial \theta_r} = \frac{\frac{\partial a}{\partial \theta_r} bc - a\left[\frac{\partial b}{\partial \theta_r} c + \frac{\partial c}{\partial \theta_r} b\right]}{(bc)^2} \tag{9.14}$$

where:

$$a = 2\alpha \sum_{k=1}^{D} \frac{\partial \mathcal{LL}_i}{\partial \theta_k} \frac{\partial \mathcal{LL}_j}{\partial \theta_k} \tag{9.15}$$

$$b = \left(1 + 2\alpha \sum_{k=1}^{D} \left(\frac{\partial \mathcal{LL}_i}{\partial \theta_k}\right)^2\right)^{1/2} \tag{9.16}$$

$$c = \left(1 + 2\alpha \sum_{k=1}^{D} \left(\frac{\partial \mathcal{LL}_j}{\partial \theta_k}\right)^2\right)^{1/2} \tag{9.17}$$

$$\frac{\partial a}{\partial \theta_r} = 2\alpha \sum_{k=1}^{D} \left[\frac{\partial^2 \mathcal{LL}_i}{\partial \theta_k \partial \theta_r} \frac{\partial \mathcal{LL}_j}{\partial \theta_k} + \frac{\partial^2 \mathcal{LL}_j}{\partial \theta_k \partial \theta_r} \frac{\partial \mathcal{LL}_i}{\partial \theta_k}\right] \tag{9.18}$$

$$\frac{\partial b}{\partial \theta_r} = 2\alpha \left(1 + 2\alpha \sum_{k=1}^{D} \left(\frac{\partial \mathcal{LL}_i}{\partial \theta_k}\right)^2\right)^{-1/2} \sum_{k=1}^{D} \frac{\partial^2 \mathcal{LL}_i}{\partial \theta_k \partial \theta_r} \frac{\partial \mathcal{LL}_i}{\partial \theta_k} \tag{9.19}$$

$$\frac{\partial c}{\partial \theta_r} = 2\alpha \left(1 + 2\alpha \sum_{k=1}^{D} \left(\frac{\partial \mathcal{LL}_j}{\partial \theta_k}\right)^2\right)^{-1/2} \sum_{k=1}^{D} \frac{\partial^2 \mathcal{LL}_j}{\partial \theta_k \partial \theta_r} \frac{\partial \mathcal{LL}_j}{\partial \theta_k} \tag{9.20}$$

9.2.4 Normalisation

Each feature / mapping combination is subject to the normalisation:

$$\hat{\mathcal{K}}(x, z) = \frac{\mathcal{K}(x, z)}{\sqrt{\mathcal{K}(x, x)\mathcal{K}(z, z)}} \tag{9.21}$$

The quotient rule can be used to ascertain the derivative:

$$\frac{\partial\left(\hat{\mathcal{K}}(x,z)\right)}{\partial\theta} = \frac{\frac{\partial f}{\partial\theta}g - f\frac{\partial g}{\partial\theta}}{g^2} \tag{9.22}$$

where:

$$f = \mathcal{K}(x,z) \tag{9.23}$$

$$g = \sqrt{\mathcal{K}(x,x)\mathcal{K}(z,z)} \tag{9.24}$$

$$\frac{\partial f}{\partial\theta} = \frac{\partial\left(\mathcal{K}(x,z)\right)}{\partial\theta} \tag{9.25}$$

$$\frac{\partial g}{\partial\theta} = \frac{1}{2}\left[\frac{\sqrt{\mathcal{K}(z,z)}}{\sqrt{\mathcal{K}(x,x)}}\frac{\partial\left(\mathcal{K}(x,x)\right)}{\partial\theta} + \frac{\sqrt{\mathcal{K}(x,x)}}{\sqrt{\mathcal{K}(z,z)}}\frac{\partial\left(\mathcal{K}(z,z)\right)}{\partial\theta}\right] \tag{9.26}$$

so that:

$$\frac{\partial\left(\hat{\mathcal{K}}(x,z)\right)}{\partial\theta} = \frac{\frac{\partial(\mathcal{K}(x,z))}{\partial\theta} - \frac{1}{2}\mathcal{K}(x,z)\left(\frac{1}{\mathcal{K}(x,x)}\frac{\partial(\mathcal{K}(x,x))}{\partial\theta} + \frac{1}{\mathcal{K}(z,z)}\frac{\partial(\mathcal{K}(z,z))}{\partial\theta}\right)}{\sqrt{\mathcal{K}(x,x)\mathcal{K}(z,z)}} \tag{9.27}$$

This means that for each mapping, the overall derivative for the normalised kernel is, using the chain rule, the product of equation (9.27) and one of (9.5), (9.8) or (9.12).

9.2.5 Autoregressive Conditional Duration model

The second order derivatives of the Autoregressive Conditional Duration model parameters are as follows:

$$\frac{\partial^2 \mathcal{LL}}{\partial \lambda \partial w} = x_t(w + qx_{t-1} + ph_{t-1})^{-2} \tag{9.28}$$
$$\frac{\partial^2 \mathcal{LL}}{\partial \lambda \partial q} = x_t x_{t-1}(w + qx_{t-1} + ph_{t-1})^{-2} \tag{9.29}$$
$$\frac{\partial^2 \mathcal{LL}}{\partial \lambda \partial p} = x_t h_{t-1}(w + qx_{t-1} + ph_{t-1})^{-2} \tag{9.30}$$
$$\frac{\partial^2 \mathcal{LL}}{\partial \lambda^2} = -\lambda^{-2} \tag{9.31}$$
$$\frac{\partial^2 \mathcal{LL}}{\partial w^2} = -2\lambda x_t(w + qx_{t-1} + ph_{t-1})^{-3} \tag{9.32}$$
$$\frac{\partial^2 \mathcal{LL}}{\partial q^2} = -2\lambda x_t x_{t-1}^2(w + qx_{t-1} + ph_{t-1})^{-3} \tag{9.33}$$
$$\frac{\partial^2 \mathcal{LL}}{\partial p^2} = -2\lambda x_t h_{t-1}^2(w + qx_{t-1} + ph_{t-1})^{-3} \tag{9.34}$$
$$\frac{\partial^2 \mathcal{LL}}{\partial q \partial w} = -2\lambda x_t x_{t-1}(w + qx_{t-1} + ph_{t-1})^{-3} \tag{9.35}$$
$$\frac{\partial^2 \mathcal{LL}}{\partial p \partial w} = -2\lambda x_t h_{t-1}(w + qx_{t-1} + ph_{t-1})^{-3} \tag{9.36}$$
$$\frac{\partial^2 \mathcal{LL}}{\partial p \partial q} = -2\lambda x_t h_{t-1} x_{t-1}(w + qx_{t-1} + ph_{t-1})^{-3} \tag{9.37}$$

9.2.6 Poisson process

Each parameter used in the Simple Poisson model's Fisher Kernel is independent of the other so only simple differentiation of (8.18) wrt λ_k for each of the levels on the Bid and Ask sides is required:

$$\frac{\partial^2 \mathcal{LL}}{\partial \theta_k^2} = -\frac{x_k}{\lambda_k^2} \tag{9.38}$$

9.2.7 Wiener distribution: $\frac{\partial^2 \mathcal{LL}}{\partial \mu^2}$

Representing the numerator and denominator from (8.30) as f and g respectively:

$$f = -\frac{\sqrt{t}}{\sigma}\phi(x) - \frac{2\alpha}{\sigma^2} z\Phi(y) + z\frac{\sqrt{t}}{\sigma}\phi(y) \tag{9.39}$$
$$g = \Phi(x) - z\Phi(y) \tag{9.40}$$

The second derivative of the log likelihood with respect to μ can be expressed:

$$\begin{aligned}\frac{\partial^2 \mathcal{LL}}{\partial \mu^2} &= \frac{\partial \left(\frac{f}{g}\right)}{\partial \mu} \\ &= \frac{\frac{\partial f}{\partial \mu} g - f \frac{\partial g}{\partial \mu}}{g^2} \\ &= \frac{\frac{\partial f}{\partial \mu} g - f^2}{g^2}\end{aligned} \tag{9.41}$$

where we still need to calculate $\dfrac{\partial f}{\partial \mu}$.

Using the expressions for the derivatives of the standardised Gaussian and cumulative Gaussian distribution functions:

$$\frac{\partial \phi\left(\mathrm{f}(x)\right)}{\partial x} = -\mathrm{f}(x) \frac{\partial \mathrm{f}(x)}{\partial x} \phi\left(\mathrm{f}(x)\right) \tag{9.42}$$

$$\frac{\partial \Phi\left(\mathrm{f}(x)\right)}{\partial x} = \phi\left(\mathrm{f}(x)\right) \frac{\partial \mathrm{f}(x)}{\partial x} \tag{9.43}$$

we can calculate the following partial derivatives:

$$\begin{aligned}\frac{\partial \left(\phi(x)\right)}{\partial \mu} &= \frac{x\sqrt{t}}{\sigma} \phi(x) \\ \frac{\partial \left(\phi(y)\right)}{\partial \mu} &= \frac{y\sqrt{t}}{\sigma} \phi(y) \\ \frac{\partial z}{\partial \mu} &= \frac{2\alpha z}{\sigma^2} \\ \frac{\partial \left(\Phi(y)\right)}{\partial \mu} &= \phi(y) \frac{\sqrt{t}}{\sigma}\end{aligned}$$

Which gives:

$$\begin{aligned}\frac{\partial f}{\partial \mu} &= -\frac{\sqrt{t}}{\sigma} \frac{\partial \left(\phi(x)\right)}{\partial \mu} - \frac{2\alpha}{\sigma^2} \frac{\partial z}{\partial \mu} \Phi(y) - \frac{2\alpha}{\sigma^2} z \frac{\partial \left(\Phi(y)\right)}{\partial \mu} + \frac{\partial z}{\partial \mu} \frac{\sqrt{t}}{\sigma} \phi(y) + z \frac{\sqrt{t}}{\sigma} \frac{\partial \left(\phi(y)\right)}{\partial \mu} \\ &= -\frac{t}{\sigma^2} x \phi(x) - \frac{4\alpha^2}{\sigma^4} z \Phi(y) + \frac{t}{\sigma^2} z y \phi(y)\end{aligned} \tag{9.44}$$

$\frac{\partial^2 \mathcal{LL}}{\partial \mu^2}$ can then be calculated by substituting (9.44), (9.39) and (9.40) into (9.41).

9.2.8 Wiener distribution: $\frac{\partial^2 \mathcal{LL}}{\partial \sigma^2}$

Now representing the numerator and denominator from (8.31) as f and g respectively:

$$f = -\frac{x}{\sigma}\phi(x) + \frac{4\mu\alpha}{\sigma^3} z\Phi(y) + z\frac{y}{\sigma}\phi(y) \tag{9.45}$$

$$g = \Phi(x) - z\Phi(y) \tag{9.46}$$

The second derivative of the log likelihood with respect to σ can be expressed:

$$\begin{aligned}\frac{\partial^2 \mathcal{LL}}{\partial \sigma^2} &= \frac{\partial\left(\frac{f}{g}\right)}{\partial \sigma} \\ &= \frac{\frac{\partial f}{\partial \sigma} g - f\frac{\partial g}{\partial \sigma}}{g^2} \\ &= \frac{\frac{\partial f}{\partial \sigma} g - f^2}{g^2}\end{aligned} \tag{9.47}$$

In a similar manner, it is useful to calculate the following partial derivatives:

$$\begin{aligned}\frac{\partial(\phi(x))}{\partial \sigma} &= \frac{x^2}{\sigma}\phi(x) \\ \frac{\partial(\phi(y))}{\partial \sigma} &= \frac{y^2}{\sigma}\phi(y) \\ \frac{\partial z}{\partial \sigma} &= -\frac{4z\mu\alpha}{\sigma^3} \\ \frac{\partial \Phi(y)}{\partial \sigma} &= -\phi(y)\frac{y}{\sigma}\end{aligned}$$

$$\begin{aligned}\frac{\partial f}{\partial \sigma} &= \frac{2x}{\sigma^2}\phi(x) - \frac{x}{\sigma}\frac{\partial(\phi(x))}{\partial \sigma} + \frac{4\mu\alpha}{\sigma^3}\left(-\frac{3}{\sigma}z\Phi(y) + \frac{\partial z}{\partial \sigma}\Phi(y) + z\frac{\partial \Phi(y)}{\partial \sigma}\right) + \frac{\partial z}{\partial \sigma}\frac{y}{\sigma}\phi(y) - \frac{2zy}{\sigma^2}\phi(y) + \frac{zy}{\sigma}\frac{\partial(\phi(y))}{\partial \sigma} \\ &= \frac{2x}{\sigma^2}\phi(x) - \frac{x^3}{\sigma^2}\phi(x) + \frac{4\mu\alpha}{\sigma^3}\left(-\frac{3}{\sigma}z\Phi(y) - \frac{4z\mu\alpha}{\sigma^3}\Phi(y) - \frac{2zy}{\sigma}\phi(y)\right) - \frac{2zy}{\sigma^2}\phi(y) + \frac{zy^3}{\sigma^2}\phi(y)\end{aligned} \tag{9.48}$$

$\frac{\partial^2 \mathcal{LL}}{\partial \sigma^2}$ can then be calculated by substituting (9.48), (9.45) and (9.46) into (9.47).

9.2.9 Wiener distribution: $\frac{\partial^2 \mathcal{LL}}{\partial \mu \partial \sigma}$

Representing the numerator and denominator from (8.30) as f and g respectively:

$$f = -\frac{\sqrt{t}}{\sigma}\phi(x) - \frac{2\alpha}{\sigma^2} z\Phi(y) + z\frac{\sqrt{t}}{\sigma}\phi(y) \tag{9.49}$$

$$g = \Phi(x) - z\Phi(y) \tag{9.50}$$

the cross derivative of the log likelihood with respect to σ and μ can be expressed:

$$\frac{\partial^2 \mathcal{LL}}{\partial \mu \partial \sigma} = \frac{\frac{\partial f}{\partial \sigma} g - f \frac{\partial g}{\partial \sigma}}{g^2} \tag{9.51}$$

$\frac{\partial g}{\partial \sigma}$ is equal to (9.45).

Making use of the partial derivatives calculated in Section 9.2.8:

$$\begin{aligned}\frac{\partial f}{\partial \sigma} &= \frac{\sqrt{t}}{\sigma^2}\phi(x) - \frac{\sqrt{t}}{\sigma}\frac{\partial(\phi(x))}{\partial \sigma} + \frac{4\alpha}{\sigma^3} z\Phi(y) - \frac{2\alpha}{\sigma^2}\frac{\partial z}{\partial \sigma}\Phi(y) - \frac{2\alpha}{\sigma^2} z\frac{\partial \Phi(y)}{\partial \sigma} + \frac{\partial z}{\partial \sigma}\frac{\sqrt{t}}{\sigma}\phi(y) - \frac{z\sqrt{t}}{\sigma^2}\phi(y) + \frac{z\sqrt{t}}{\sigma}\frac{\partial(\phi(y))}{\partial \sigma} \\ &= \frac{\sqrt{t}}{\sigma^2}\phi(x) - \frac{\sqrt{t}}{\sigma^2}x^2\phi(x) + \frac{4\alpha z}{\sigma^3}\Phi(y) + \frac{8\alpha^2 z\mu}{\sigma^5}\Phi(y) + \frac{2\alpha z y}{\sigma^3}\phi(y) - \frac{4z\mu\alpha\sqrt{t}}{\sigma^4}\phi(y) - \frac{z\sqrt{t}}{\sigma^2}\phi(y) + \frac{z\sqrt{t}y^2}{\sigma^2}\phi(y)\end{aligned} \tag{9.52}$$

$\frac{\partial^2 \mathcal{LL}}{\partial \mu \partial \sigma}$ can then be calculated by substituting (9.52), (9.49), (9.50) and (9.45) into (9.51).

10 Experimental plan

10.1 Motivation

The principal aim of this research is to investigate the marriage of machine learning techniques with models from the financial domain. With this in mind, we wish to discover whether discriminative machine learning techniques can be informative regarding the effectiveness of standard order book based features and price based technical analysis methods. In the context of Multiple Kernel Learning, this translates into understanding which kernel mappings and features are the most effective at making predictions on an individual basis and also whether MKL selects these kernels and allocates them significant weightings when they form part of an overall set.

Following this initial area and the questions it raises, there is the issue of the effectiveness of incorporating generative models, embodied as Fisher kernels, into the discriminative framework and understanding their usefulness in our predictive ambitions.

Finally, we wish to investigate whether allowing hyperparameter learning and learning of the parameters in the generative models is useful or whether it is taking things too far to learn the set of kernels as well as the kernels themselves.

Breaking these three main areas of interest down into some simple questions that can be answered experimentally, the experiments need to be designed so that they address the following more specific questions:

10.1.1 General kernel selection

Which kernel mapping / features combinations are the best performers when used individually?

The majority of the literature points to the RBF mapping being the most effective in terms of performance for financial prediction problems and it will be interesting to see if the experiments show that out of the three mappings being investigated (RBF, Polynomial and INN) it is indeed the RBF that is consistently more effective than the others. Given that there is no evidence of previous research on using

either the volumes in order book data or Fisher kernels to forecast price movements, it will be of use to see which class of features is the most effective: price action based, order book volume based or those based on Fisher kernels. From the literature, one might expect the efficacy of the three classes to vary across the different time horizons, with order book and Fisher based features to be more effective over the shorter horizons and price based ones over the longer and the experiments will highlight whether this is the case. There may also be one particular feature that stands out overall in its performance that will exemplify its corresponding trading rule, volume feature or MM based model.

How do the individual kernels compare to a simple benchmark and how stable is their performance over time?

It may be that the individual kernels are actually less successful at making predictions than a naive predictor on average or that their outpeformance is so slight as to not justify the increase in computational complexity they entail. Furthermore, it may also be that the stability in their performance over time (i.e. over all the out-of-sample periods) is so low as to render them unreliable.

10.1.2 Standard MKL

Which combinations are awarded the highest weightings by standard MKL?

One might expect that the kernels awarded the highest weightings are also those that are selected the most frequently, and this is something that we will investigate. However, it may be that the stability of these weightings and selections is so low that meaningful conclusions cannot be drawn in this area. Is there a particular kernel mapping, feature or class of feature which is consistently awarded the highest weightings / selected the most often? Another line of inquiry is whether the kernels with the highest weightings / most often selected are also the ones that perform the best on an individual basis. Are there some kernels that show poor performance when used individually, but seem consistently to be selected during MKL? It will also be interesting to see what the performance of the average kernel (i.e. allocating each kernel in the set the same weight) is.

What happens when subsets of the full MKL set are used

It will be interesting to see the impact of using subsets of the full kernel set. The two subsets which will be the most informative will be one based on the most commonly selected kernels from the full standard MKL run and one based on the most effective of the kernels when they are run individually. The performance comparison between these two subsets and the full set will give us information about whether MKL seems occasionally to be selecting kernels that appear to be fairly low performing when taken individually, but add value when added to other kernels and also about how effective MKL is in general with larger sets. Is MKL actually able to effectively distinguish between kernels when the set is large or does reducing the size of the set it can choose from greatly improve its performance?

10.1.3 Kernel learning

Does learning the kernel improve performance?

We shall investigate the impact on performance of extending MKL so that not only are kernel weightings learnt from the data but also aspects of the kernels themselves. Will the predictive ability increase when the kernel hyperparameters and the Fisher feature parameters are learnt or will this be taking things too far and lead to significant overfitting?

Remembering the definition of β from Section 9.1 and that it is this value that the kernel learning process is attempting to minimise, we can assess the extent that kernel learning has taken place from a theoretical perspective before any predictions are made. We can therefore investigate whether there is a relationship between the extent of kernel learning as measured through β and the resulting accuracy of the predictor.

Also, from a kernel learning perspective we shall be able to identify what the optimal hyperparameter values / Fisher feature parameters are and contrast them with the ones that proved most effective on an individual basis (for the hyperparameters) and the ML values (for the Fisher feature parameters).

Lastly, it will be useful to see if there is a significant difference in performance between the parallel and serial optimisation methods used for kernel learning and whether the former's outperformance justifies its increased computational complexity (and mathematical effort). One particular area of interest will be the

relationship between the extent of optimisation (as represented by reduction in β) and the performance that is borne out by the predictor. Does a greater reduction in β mean better performance or does it just mean further overfitting?

10.1.4 Practical considerations

It will be interesting to see whether the increased complexity that MKL and kernel learning entail is justified through increased performance and also whether the complexity of some of the techniques is so significant as to render them implasuble for real time applications.

A further issue of importance from a pragmatic perspective is how frequently the different techniques are able to make predictions - do the methods that have a significantly high predictive accuracy only make their predictions infrequently - i.e. is there a trade-off between predictive accuracy and frequency?

10.2 Kernel sets investigated

10.2.1 Full kernel set

A set $\mathcal{F}$ consisting of each of the 3 Fisher kernels described in (8.14), (8.21) and (8.32) along with the feature set described in Section 7.1 will be constructed:

$$\mathcal{F}_1 = \left\{ EMA_t^{L_1}, \ldots, EMA_t^{L_N} \right\}$$

$$\mathcal{F}_2 = \left\{ MA_t^{L_1}, \ldots, MA_t^{L_N}, \sigma_t^{L_1}, \ldots, \sigma_t^{L_N} \right\}$$

$$\mathcal{F}_3 = \left\{ P_t, \max_t^{L_1}, \ldots, \max_t^{L_N}, \min_t^{L_1}, \ldots, \min_t^{L_N} \right\}$$

$$\mathcal{F}_4 = \left\{ \Uparrow_t^{L_1}, \ldots, \Uparrow_t^{L_N}, \Downarrow_t^{L_1}, \ldots, \Downarrow_t^{L_N} \right\}$$

$$\mathcal{F}_{5\ldots8} = \left\{ \boldsymbol{V}_t, \frac{\boldsymbol{V}_t}{\|\boldsymbol{V}_t\|_1}, \boldsymbol{V}_t - \boldsymbol{V}_{t-1}, \frac{\boldsymbol{V}_t - \boldsymbol{V}_{t-1}}{\|\boldsymbol{V}_t - \boldsymbol{V}_{t-1}\|_1} \right\}$$

$$\mathcal{F}_{9\ldots11} = \left\{ \mathbf{g}_{\mathbf{ACD}}, \mathbf{g}_{\mathbf{Poiss}}, \mathbf{g}_{\mathbf{Wiener}} \right\}$$

A feature mapping set consisting of the kernel types described in section 7.2 (without the linear kernel) will be used:

$$\mathcal{K}_{1:5} = \left\{ \exp\left(- \left\| \boldsymbol{x} - \boldsymbol{x}' \right\|^2 / \sigma_1^2 \right), \ldots, \exp\left(- \left\| \boldsymbol{x} - \boldsymbol{x}' \right\|^2 / \sigma_5^2 \right) \right\}$$

$$\mathcal{K}_{6:10} = \left\{ \left(\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + 1 \right)^{d_1}, \ldots, \left(\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + 1 \right)^{d_5} \right\}$$

$$\mathcal{K}_{11:15} = \left\{ \frac{2}{\pi} \sin^{-1}\left(\frac{2\boldsymbol{x}^T \Sigma_1 \boldsymbol{x}'}{\sqrt{(1+2\boldsymbol{x}^T \Sigma_1 \boldsymbol{x})(1+2\boldsymbol{x}'^T \Sigma_1 \boldsymbol{x}')}} \right), \ldots, \frac{2}{\pi} \sin^{-1}\left(\frac{2\boldsymbol{x}^T \Sigma_5 \boldsymbol{x}'}{\sqrt{(1+2\boldsymbol{x}^T \Sigma_5 \boldsymbol{x})(1+2\boldsymbol{x}'^T \Sigma_5 \boldsymbol{x}')}} \right) \right\}$$

This means that altogether there will be $|\mathcal{F}| \times |\mathcal{K}| = 11 \times 15 = 165$ individual kernels along with an average kernel constructed by taking the average of the 165 kernel matrices, giving 166 individual kernels altogether[4].

10.2.2 Reduced kernel set

In order to investigate what happens when the MKL set is reduced to a subset based on the most commonly selected or most effective from the full set, two re-

[4]Note that the average kernel is only used as a benchmark and is not included in the set of kernels used for combination by MKL.

duced kernel sets will be used:

Selection based on Stability / Highest Weightings (Reduced Set A)
A reduced set will be formed by investigating the weightings from a standard MKL run and taking the N_A most commonly selected individual kernels along with the N_A kernels with the most significant weightings for each of the six time horizons.

Selection based on Efficacy (Reduced Set B)
A reduced set will be formed by taking the N_B most effective individual SVM for each of the six time horizons.

N_A and N_B will be selected to give roughly the same size subset in both cases (which will not simply be $N \times 6$ because of overlap, for example because some kernels may be allocated high weightings for multiple time horizons etc).

10.3 Methodology

The general experimental procedure, as described in Section 6.4 (without the SimpleMKL / LPBoostMKL comparison), will again be adopted and repeated in the following 16 sets of experiments:

Table 10: Experiment sets

Experiment Name	**Kernel Set**	**Kernel Learning**	
		Hyperparameter	*Fisher Parameter*
Individual	Individual SVM	None	None
Standard	MKL Full Set	None	None
Serial	MKL Full Set	Serial	Serial
Parallel	MKL Full Set	Parallel	Parallel
SerialF	MKL Full Set	None	Serial
ParallelF	MKL Full Set	None	Parallel
StandardReduxA	MKL Subset A	None	None
SerialReduxA	MKL Subset A	Serial	Serial
ParallelReduxA	MKL Subset A	Parallel	Parallel
SerialReduxFA	MKL Subset A	None	Serial
ParallelReduxFA	MKL Subset A	None	Parallel
StandardReduxB	MKL Subset B	None	None
SerialReduxB	MKL Subset B	Serial	Serial
ParallelReduxB	MKL Subset B	Parallel	Parallel
SerialReduxFB	MKL Subset B	None	Serial
ParallelReduxFB	MKL Subset B	None	Parallel

The predictive accuracy of each feature / kernel combination will be assessed in terms of the percentage of possible predictions (as described in section 6.4) and the percentage of correct predictions.

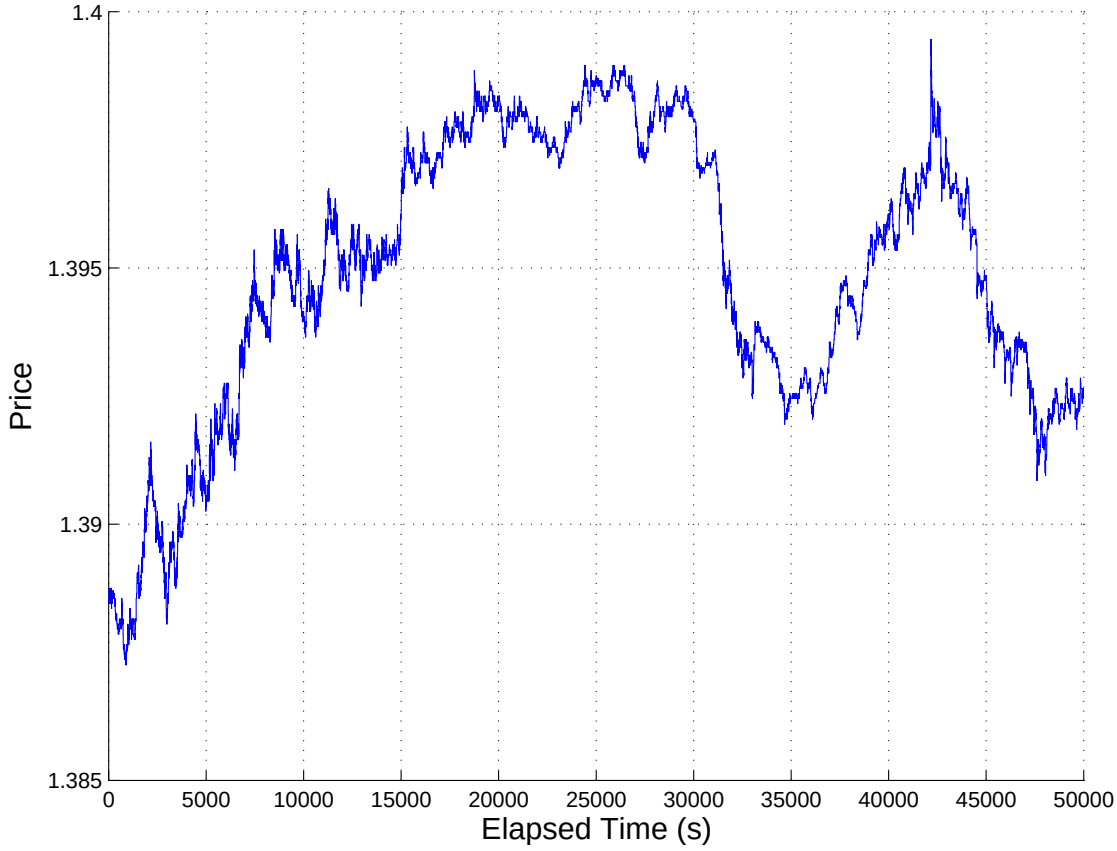
10.4 Benchmark

These results will be compared with a very simple benchmark, namely a classifier which predicts for the entire out-of-sample region, whatever the most common class was in the in-sample period. For example, if the price was mostly moving up in the in-sample period and had hence been assigned '+1' in the majority of cases, then this benchmark classifier would assign '+1' to the entire out-of-sample period. This benchmark embodies both the trend-following technical indicator and the principle of a random classifier (which is significantly more effective than simply predicting each class with a probability of a third).

11 Experimental results

Figure 7 shows the price of EURUSD over the period in question.

Figure 7: Price of EURUSD over dataset



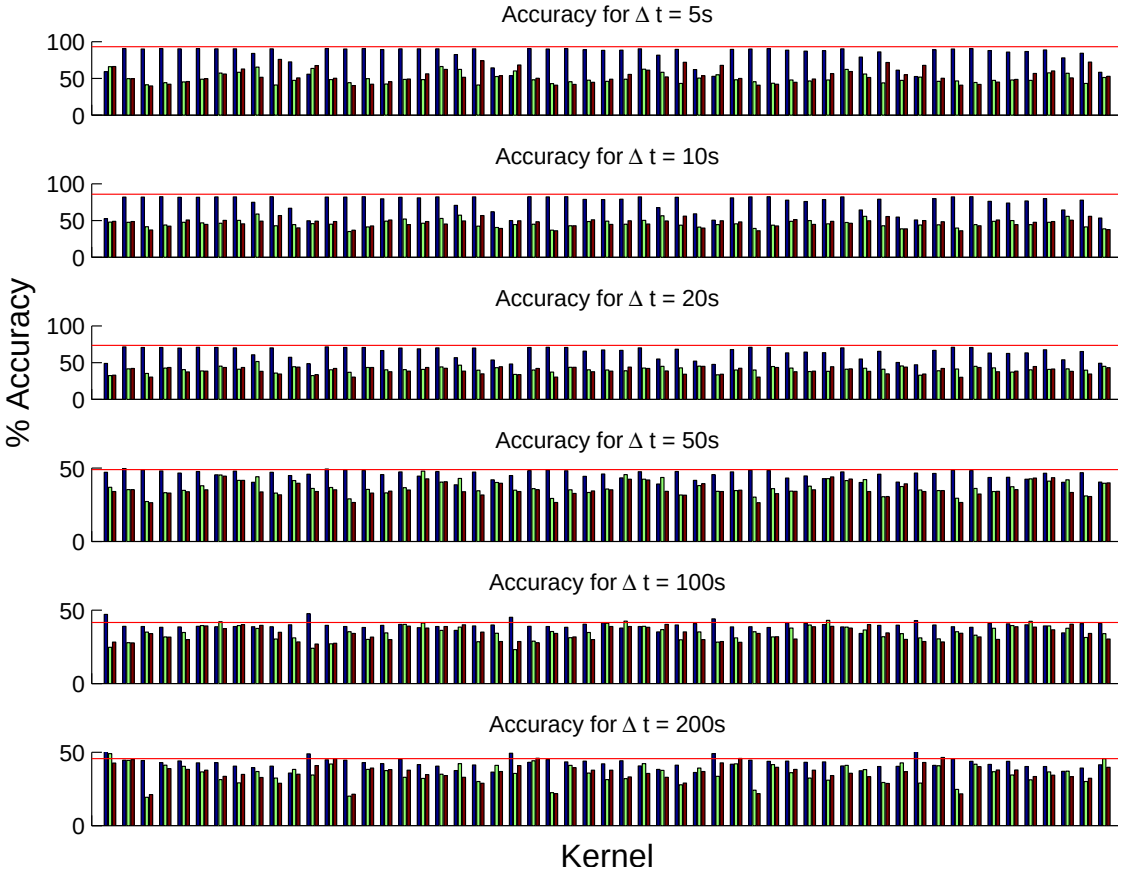
This dataset was chosen because it encompasses trends of varying severity in both directions as well as some range trading characteristics.

11.1 General kernel selection

11.1.1 Accuracy comparison

In order to investigate which kernel mapping / feature combinations were the best performers when used individually, the percentage accuracy of each out-of-sample period was calculated over the different predictive time horizons. Figure 8 shows the average of this accuracy for all of the 165 kernels for each of the horizons along with the performance of the benchmark (shown as the horizontal red line). One can see that the benchmark outperforms all the individual SVM for shorter time horizons and the majority of them for the longer horizons.

Figure 8: % accuracy of individual SVM against benchmark



With the aim of establishing whether a particular feature, kernel mapping or class of feature (i.e. price based, volume based or Fisher feature) was particularly effective, these results are summarised to highlight these different ways of grouping the 165 kernels and shown in figures 9, 10 and 11. The actual numbers for these

graphs can be found in the corresponding Tables 15, 16 and 17 in the appendix. *Refer to Section 10.2.1 for a reminder of what each of the features F_1 to F_{11} are.*

Figure 9: % accuracy of individual SVM by feature type

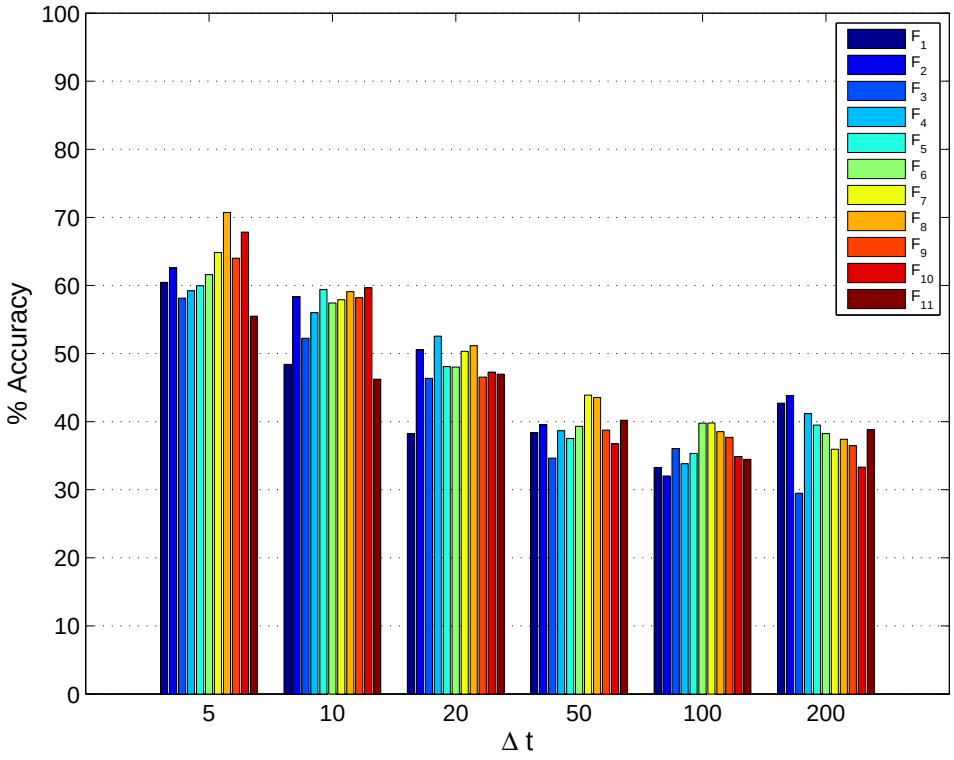


Figure 10: % accuracy of individual SVM by kernel mapping

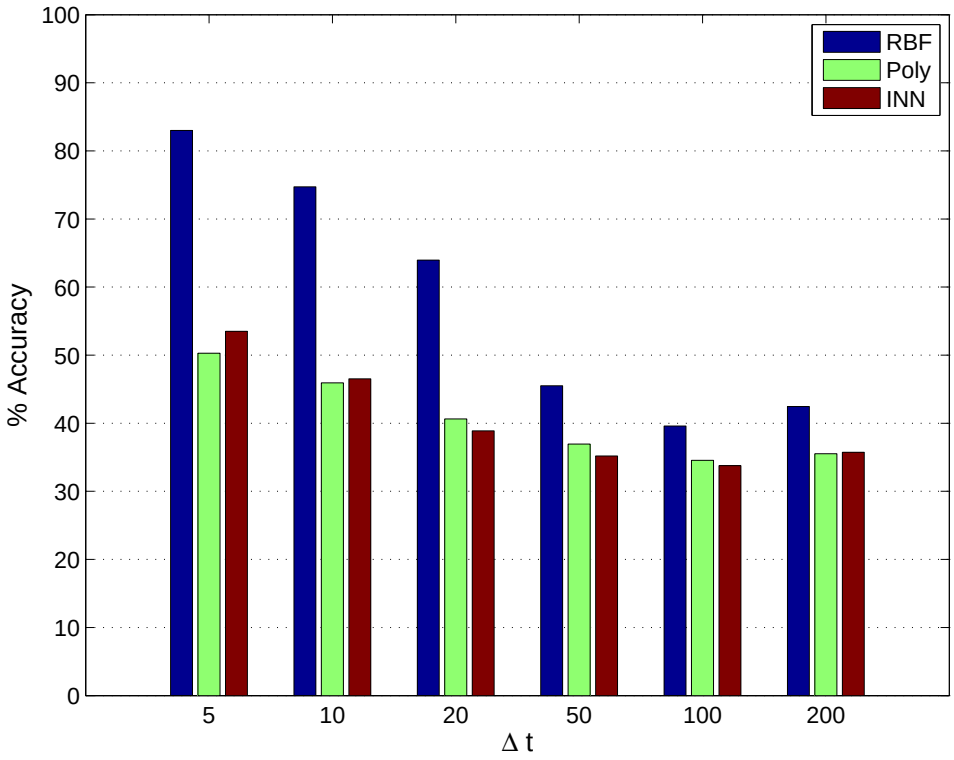
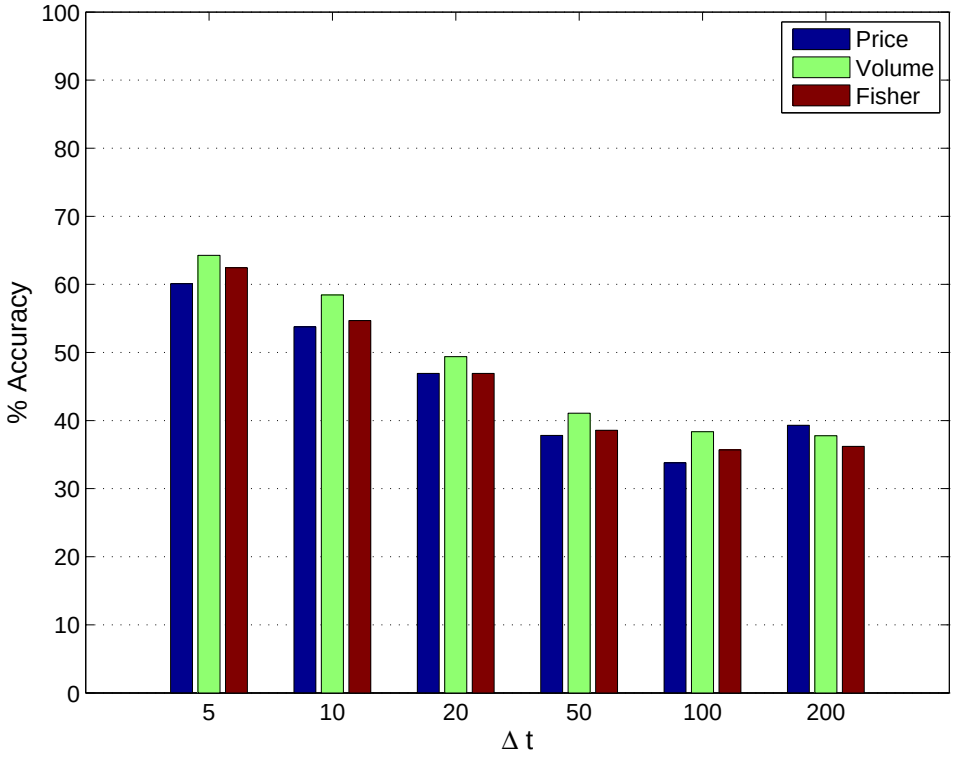


Figure 11: % accuracy of individual SVM by feature class



Looking at Figure 9, one can see that there is little consistency in the relative performance of the features across the different predictive time horizons with F_8 having the highest predictive accuracy for the shortest time horizon $\Delta t = 5s$, F_{10} for $\Delta t = 10s$, F_4 for $\Delta t = 20s$, F_7 for $\Delta t = 50s$ & $\Delta t = 100s$ and F_2 for $\Delta t = 200s$.

This lack of consistency is not evident when examining the kernel mappings. Figure 10 shows the RBF mapping outperforming the Poly and INN mappings for all time horizons, significantly so for the shorter ones.

One can see that that the relative performance of the different feature classes is also very consistent, with for time horizons $\Delta t \leq 100$, the volume based features perform most effectively and the Fisher based ones coming in second place. This is not the case for the longest time horizon, where price based features are the most effective.

The figures in this section show that percentage accuracy decreases with further time horizons until $\Delta t = 100$ but then increases slightly for $\Delta t = 200$.

11.1.2 Stability comparison

Figure 12: Time series % accuracy comparing best individual SVM to benchmark

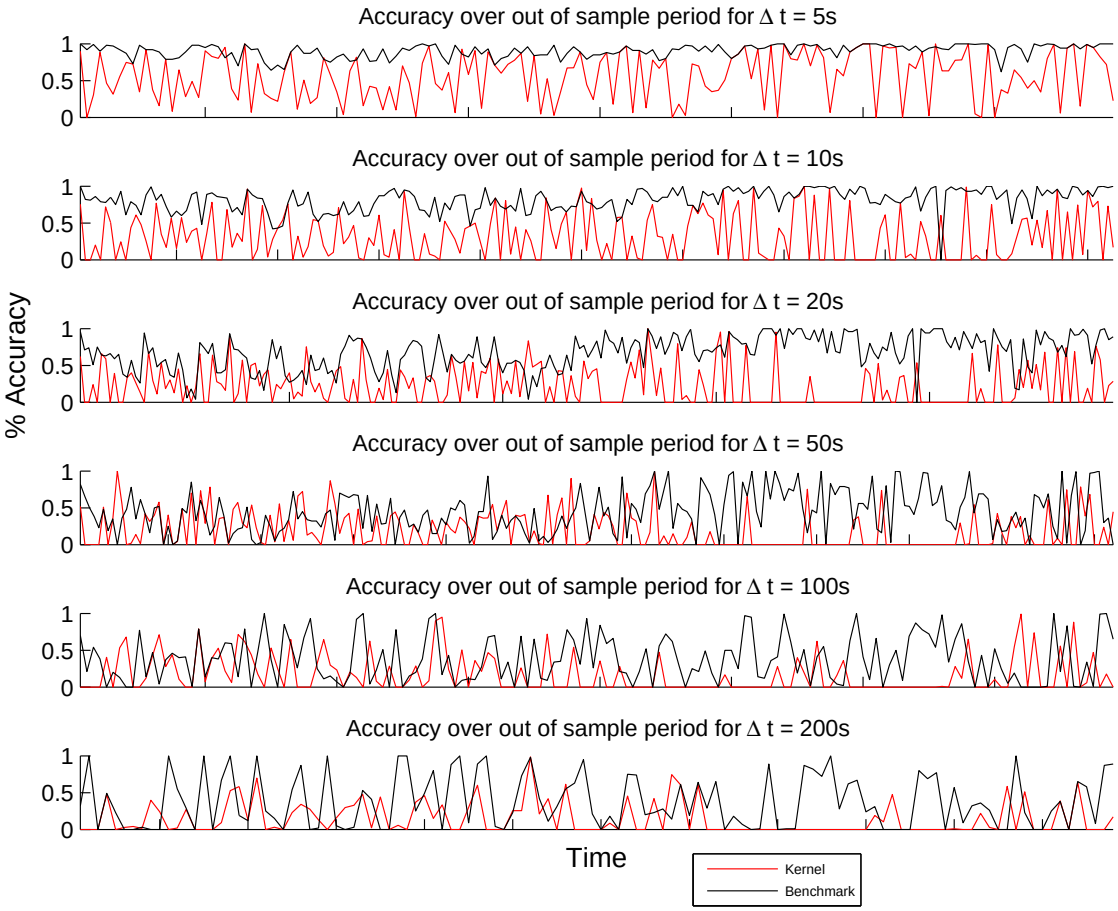


Figure 12 compares the percentage accuracy of the best individual SVM as a time series against that of the benchmark (as described in Section 10.4). Once can see that for the shorter time horizons, the benchmark's performance is much more consistent than that of the best individual SVM, but its stability decreases to be similar to that of the best individual kernels with increased predictive horizon.

In order to calculate the stability in predictive ability, the variance in the percentage accuracy was calculated across the out-of-sample periods for each predictor. The inverse of the variance, which I will call the stability, is shown for all 165 individual SVM in Figure 39 in the appendix.

Figures 13, 14 and 15 shows these stabilities summarised by feature type, kernel

mapping and feature class. See Tables 18, 19 and 20 in the appendix for the actual numbers.

Figure 13: Stability of individual SVM by feature type

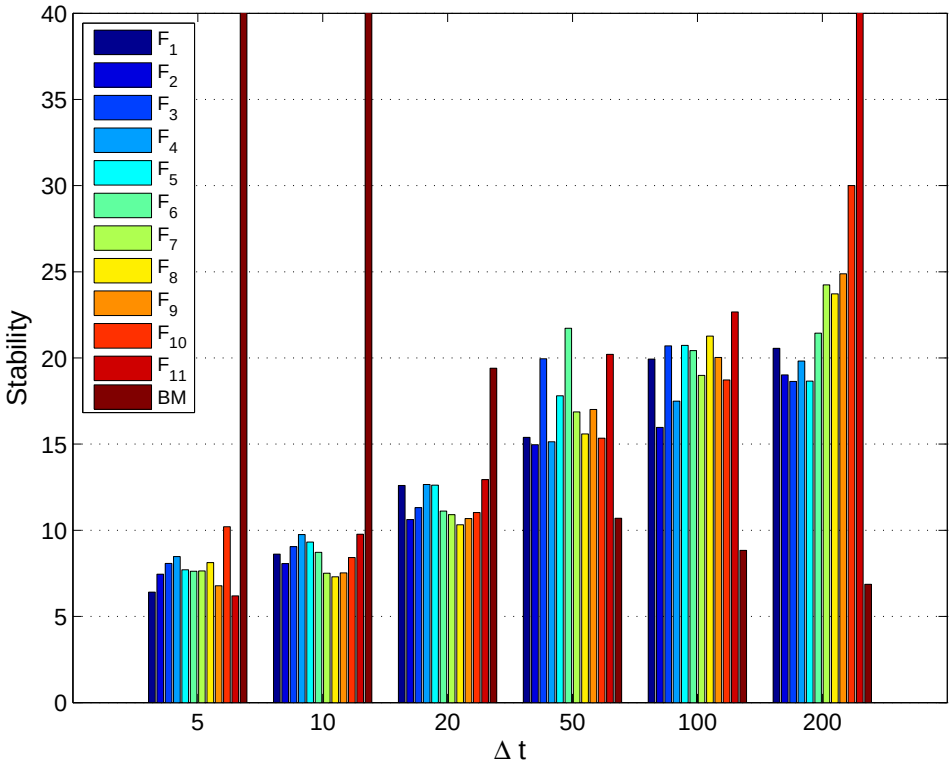


Figure 14: Stability of individual SVM by kernel mapping

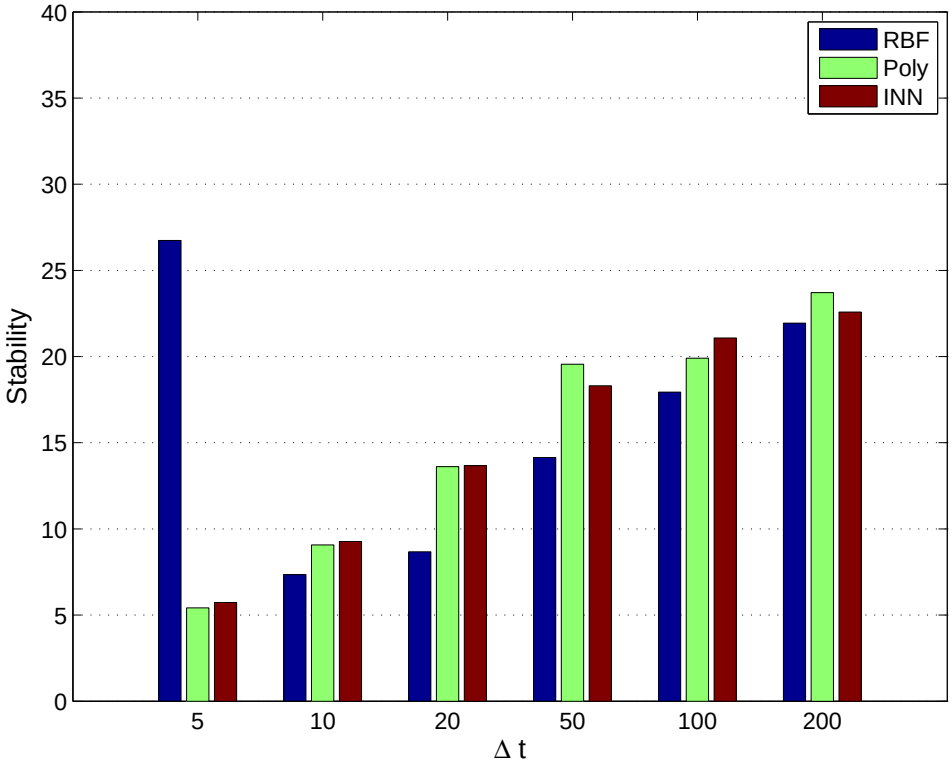
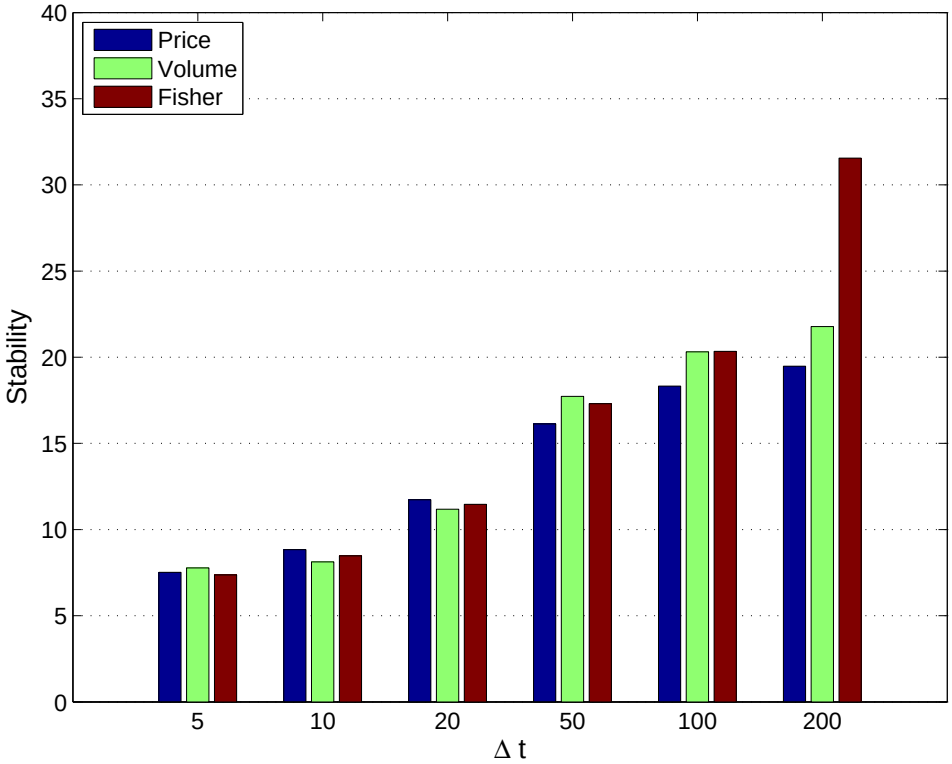


Figure 15: Stability of individual SVM by feature class



One can see from Figure 13 that other than $\Delta t = 50$, F_{11} has the highest stability out of the features for all time horizons. The benchmark has significantly greater stability than any of the features for $\Delta t \leq 20$, but significantly lower for the longer predictive horizons.

Figure 14 shows that for $\Delta t \leq 20$, the INN is generally the kernel mapping with the highest stability, followed very closely by the Polynomial kernel mapping. This is switched round for the longer time horizons, but in all cases $\Delta t > 5$ the RBF is the least stable predictor.

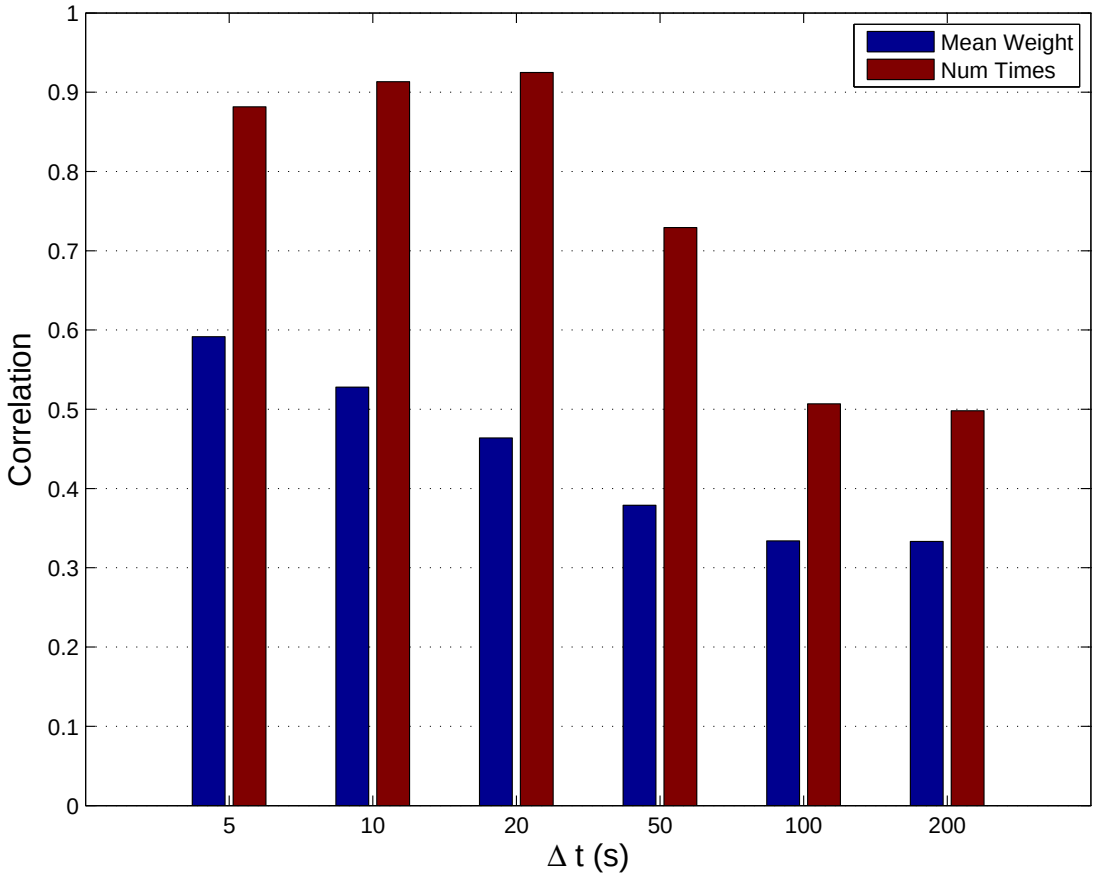
In terms of class of feature, Figure 15 shows that the classes are fairly similar in their stability at all time horizons other than $\Delta t = 200$, where the Fisher feature class is significantly more stable than the other two classes.

The figures in this section show that the stability of the predictors increases with time horizon whilst that of the benchmark decrease dramatically.

11.2 Standard MKL

Figure 40 in the appendix shows a scatter-plot of the relationship between the weighting a kernel was allocated on average by MKL and its overall percentage accuracy (Top) and the number of times it was selected by MKL and its overall percentage accuracy (Bottom). The corresponding correlation coefficients for these relationships are shown in Figure 16 (Table 21 in the appendix).

Figure 16: Correlation of MKL average weighting & number of times included vs % accuracy



One can see that the number of times a kernel is included by MKL on average has a strong relationship with how accurate that kernel is when it is used as a predictor on its own. There is also a similar relationship with the average weighting each kernel is awarded and its corresponding accuracy; however, it is less strong. In both cases, the relationship generally decreases in strength with predictive horizon.

Figure 41 in the appendix shows the average weighting that MKL allocated each

of the individual SVM over the different time horizons. In order to understand whether there is a particular kernel mapping, feature or class of feature which is consistently awarded the highest weightings, Figures 17, 18 and 19 show these weightings summarised in these three different manners. The actual numbers are shown in Tables 22, 23 and 24 in the appendix.

Figure 17: Average weighting of individual SVM by feature type

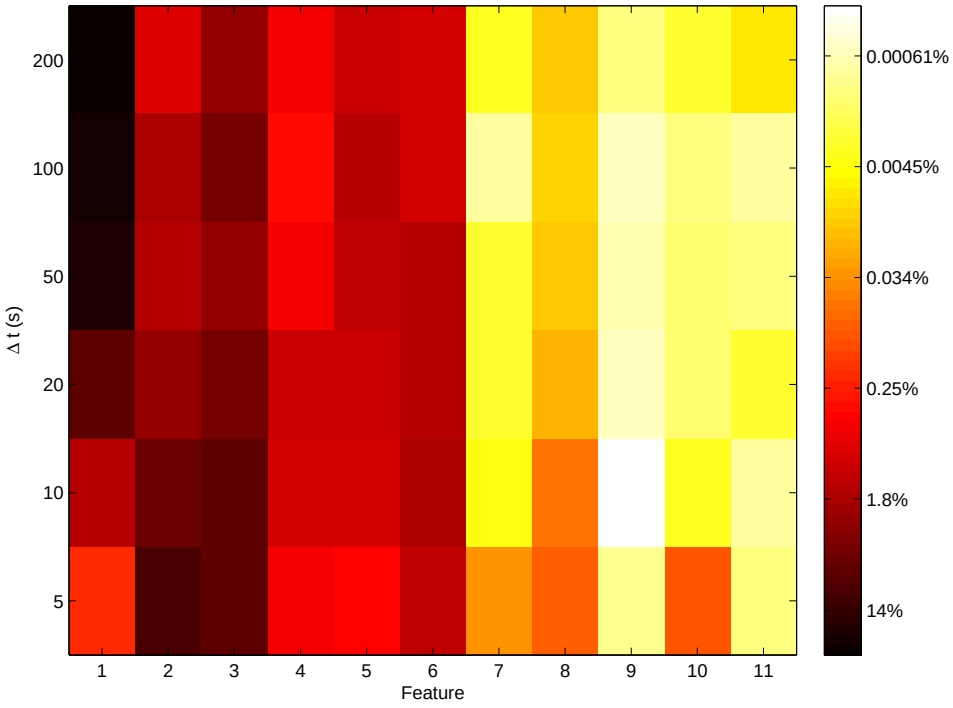


Figure 18: Average weighting of individual SVM by kernel mapping

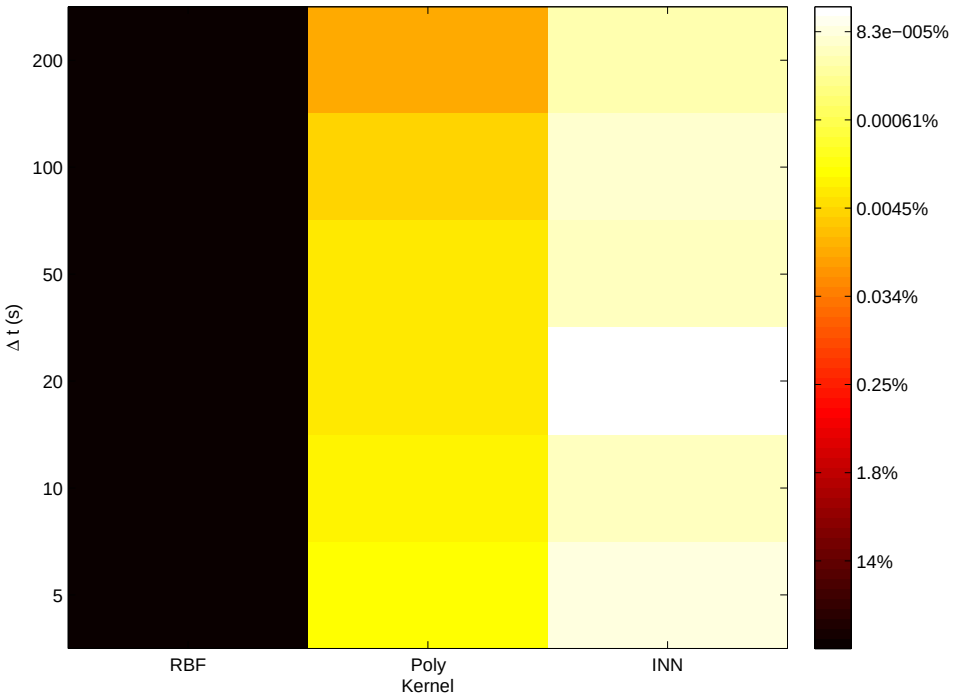


Figure 19: Average weighting of individual SVM by feature class

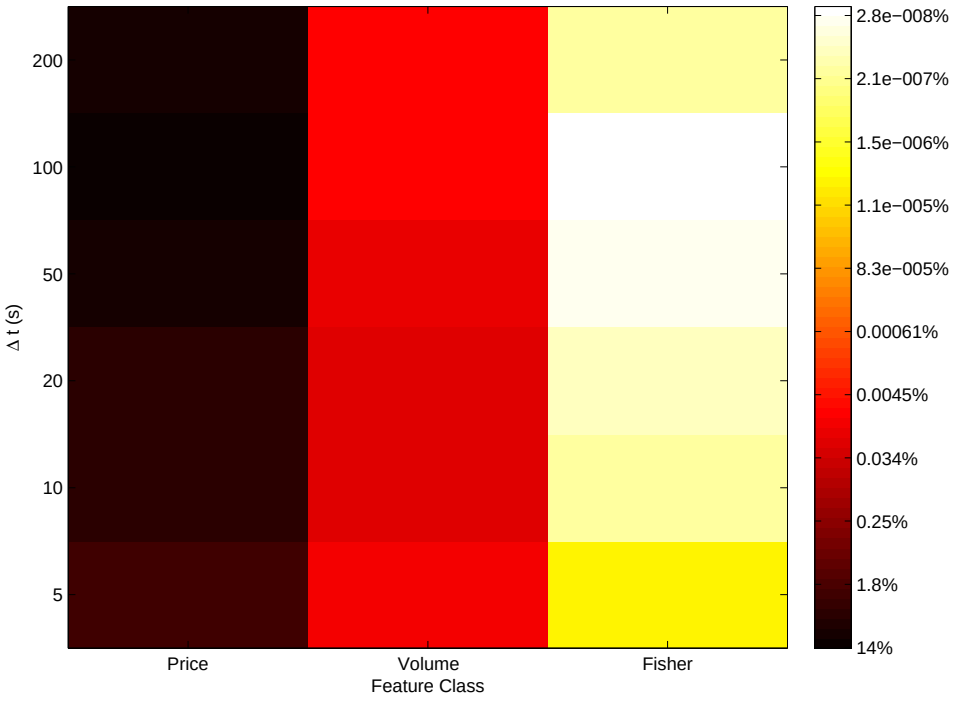


Figure 17 indicates that the first 6 features (F_1 to F_6) are awarded significantly higher weightings on average than the others. One can also see that the RBF

kernel mapping is allocated much higher weightings than the Polynomial and INN mappings from Figure 18. Furthermore, Figure 19 shows that price based features are favoured by MKL significantly more than volume or Fisher based ones, the latter being awarded very low weightings. In all cases, there is a great deal of consistency in these weightings across the different time horizons.

11.2.1 Kernel subset

Selection based on Stability / Highest Weightings (Reduced Set A)
Using $N_A = 3$, the overall set contained 7 members instead of 18 because many of the kernels were both selected frequently and allocated high weightings for multiple time horizons. These 7 kernels were as follows:

- F_1RBF_1 EMA Price based feature
- F_2RBF_1 [SMA SD] Price based feature
- F_3RBF_1 [Mins Maxs] Price based feature
- F_3RBF_2 [Mins Maxs] Price based feature
- F_3RBF_5 [Mins Maxs] Price based feature
- F_5RBF_5 Volume based feature
- F_6RBF_5 WeightedVolumes Volume based feature

Selection based on Efficacy (Reduced Set B)
Using $N_B = 2$, the overall set contained 8 members instead of 12 because some of the individual kernels were the best predictors for multiple time horizons. These 8 kernels were as follows:

- F_1RBF_1 EMA Price based feature
- F_2RBF_1 [SMA SD] Price based feature
- F_4RBF_1 [Ups Downs] Price based feature
- F_6RBF_1 WeightedVolumes Volume based feature
- $F_{10}RBF_1$ Poisson Fisher feature
- F_1RBF_2 EMA Price based feature
- F_2RBF_2 [SMA SD] Price based feature
- F_1RBF_5 EMA Price based feature

Figures 20 and 21 (Tables 25 and 26 in the appendix) compare the percentage accuracy and stability of the benchmark, average kernel, the full MKL set and subsets A and B.

Figure 20: % accuracy of benchmark, average kernel, full MKL set and Subsets A & B

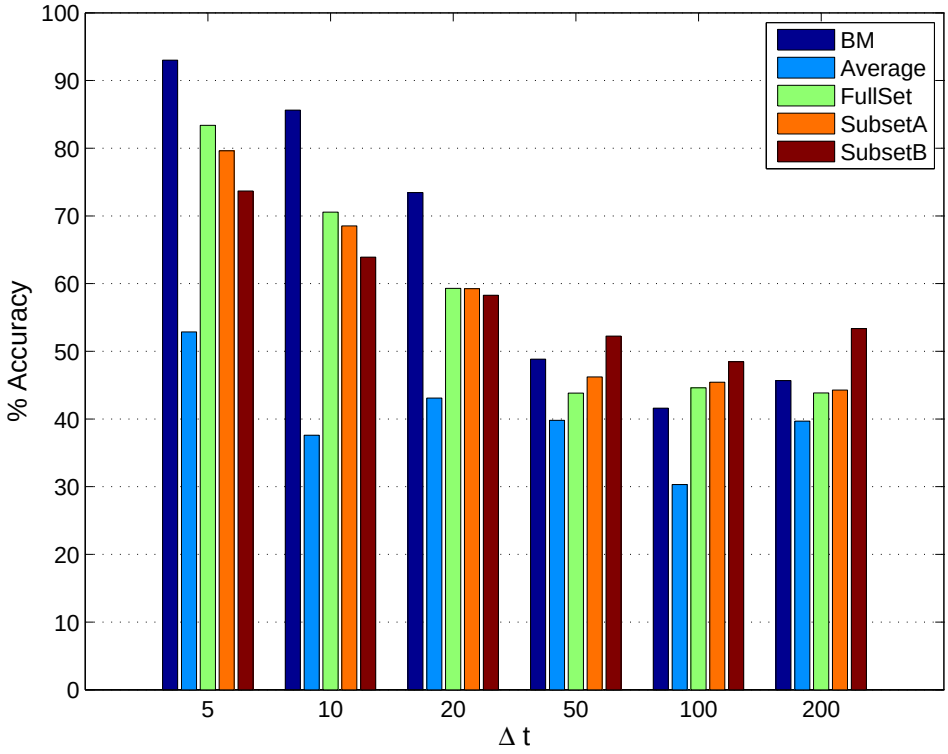


Figure 21: Stability of benchmark, average kernel, full MKL set and Subsets A & B

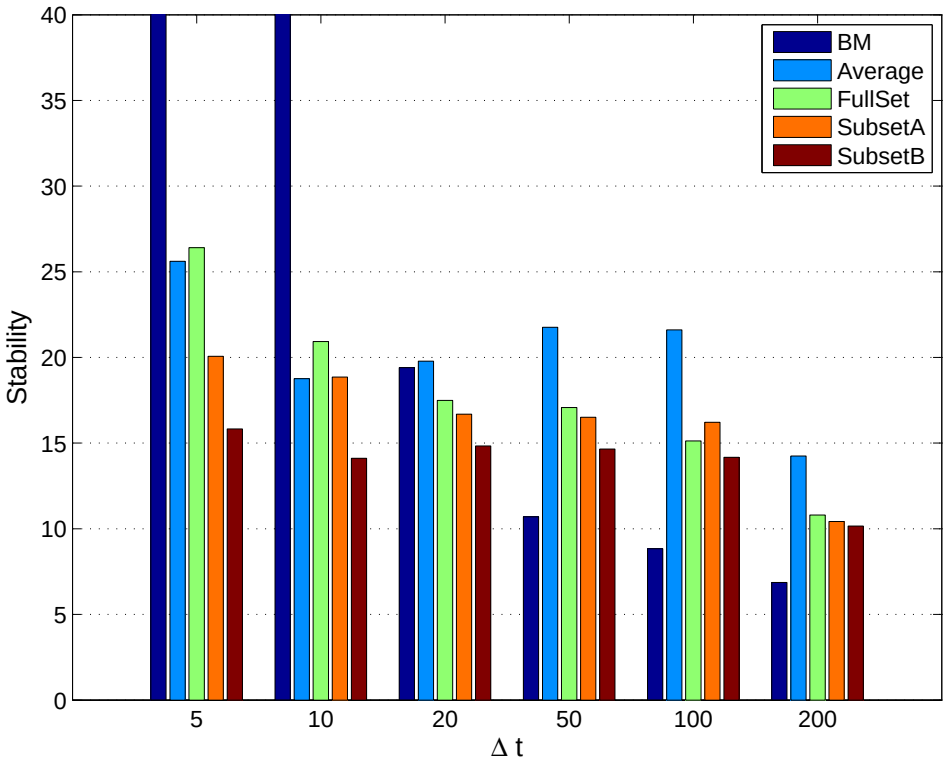


Figure 20 shows that the percentage accuracy of both subsets is better than the full MKL set for the longer time horizons ($\Delta t \geq 50$), but worse for the shorter ones. Subset A outperforms Subset B for the shorter time horizons, but this is reversed for longer ones. The BM outperforms all other methods for shorter time horizons, but is generally worse for longer ones. The performance of the average kernel is worse than all the other methods over all time horizons.

Once can see in Figure 21 that the stability of the different sets is fairly similar, with the full MKL set being generally the most stable in its performance. The benchmark is significantly more stable than the other methods for lower predictive time horizons, but significantly less stable for longer ones. The average kernel is the most stable predictor for the mid to longer time horizons ($\Delta t \geq 20$).

Figures 22 and 23 (Tables 27 and 28 in the appendix) show the average weighting each of the kernels in subsets A and B were allocated for the different time horizons.

Figure 22: Average weighting for Subset A

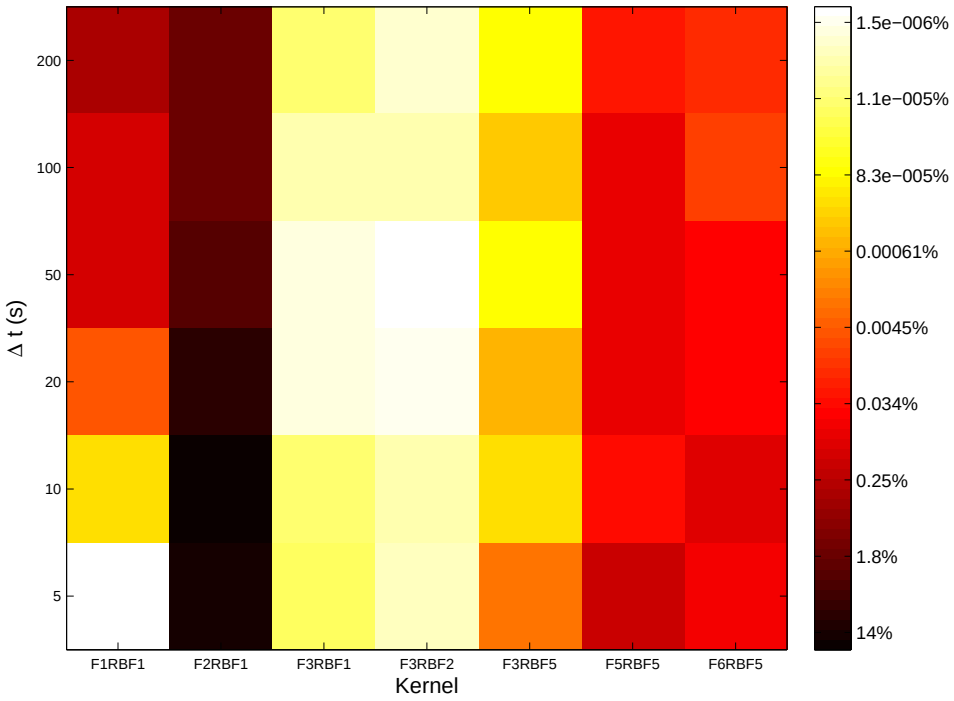


Figure 23: Average weighting for Subset B

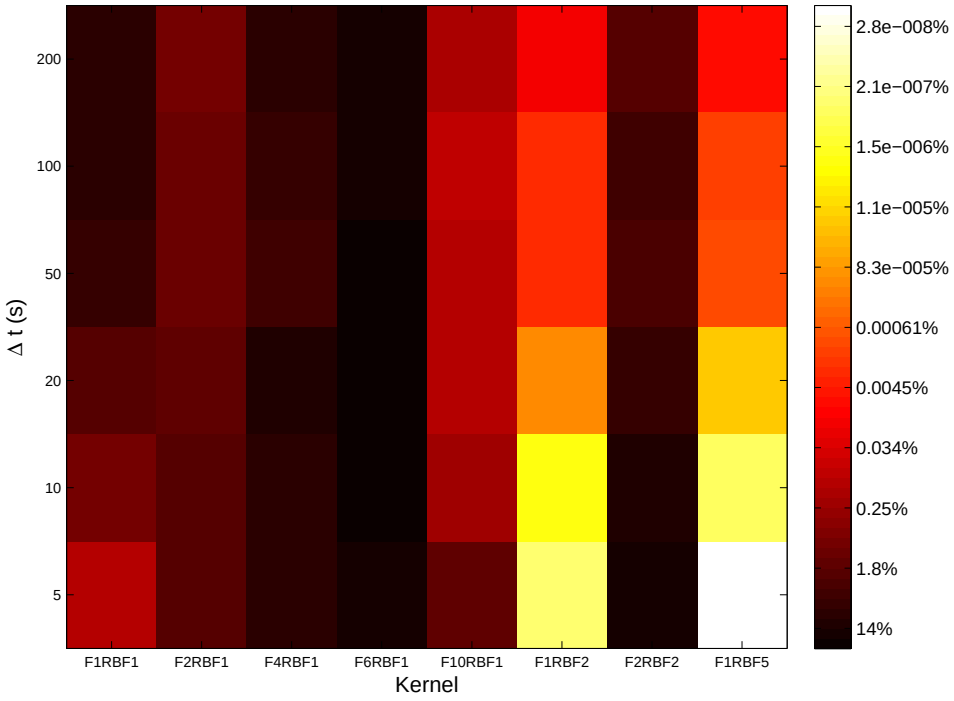


Figure 22 shows that for Subset A, the only kernel to receive above 20% weighting on average across all time horizons is F_2RBF_1. The weightings for Subset

B are slightly more equally distributed, as Figure 23 shows, with all but kernels $F_{10}RBF_1$, F_1RBF_2 and F_1RBF_5 getting $> 10\%$ weightings in the majority of cases.

11.3 Kernel learning

11.3.1 Reduction in β

Figures 24, 25 and 26 (Tables 29, 30 and 31 in the appendix) compare the percentage accuracy of the kernel learning methods for the full MKL set, Subset A and Subset B. The following terms are used in the legends of these figures:

- *No Learning* - denoting that no kernel learning (of either kernel mapping hyperparameters or Fisher feature parameters) has taken place.
- *Serial Hyper. and Fish.* - denoting that both kernel mapping hyperparameter and Fisher feature parameter learning has taken place using the serial optimisation method.
- *Parallel Hyper. and Fish.* - denoting that both kernel mapping hyperparameter and Fisher feature parameter learning has taken place using the parallel optimisation method.
- *Serial Fish.* - denoting that only Fisher feature parameter learning has taken place using the serial optimisation method.
- *Parallel Fish.* - denoting that only Fisher feature parameter learning has taken place using the parallel optimisation method.

Figure 24: Comparison of % accuracy between kernel learning methods for full MKL set

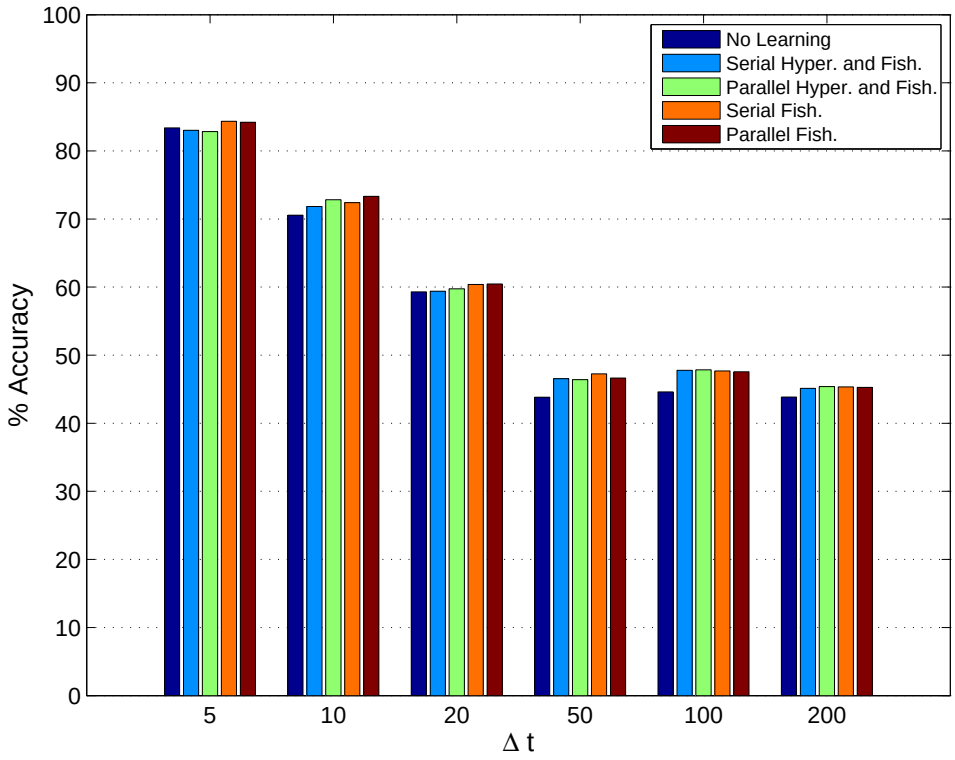


Figure 25: Comparison of % accuracy between kernel learning methods for Subset A

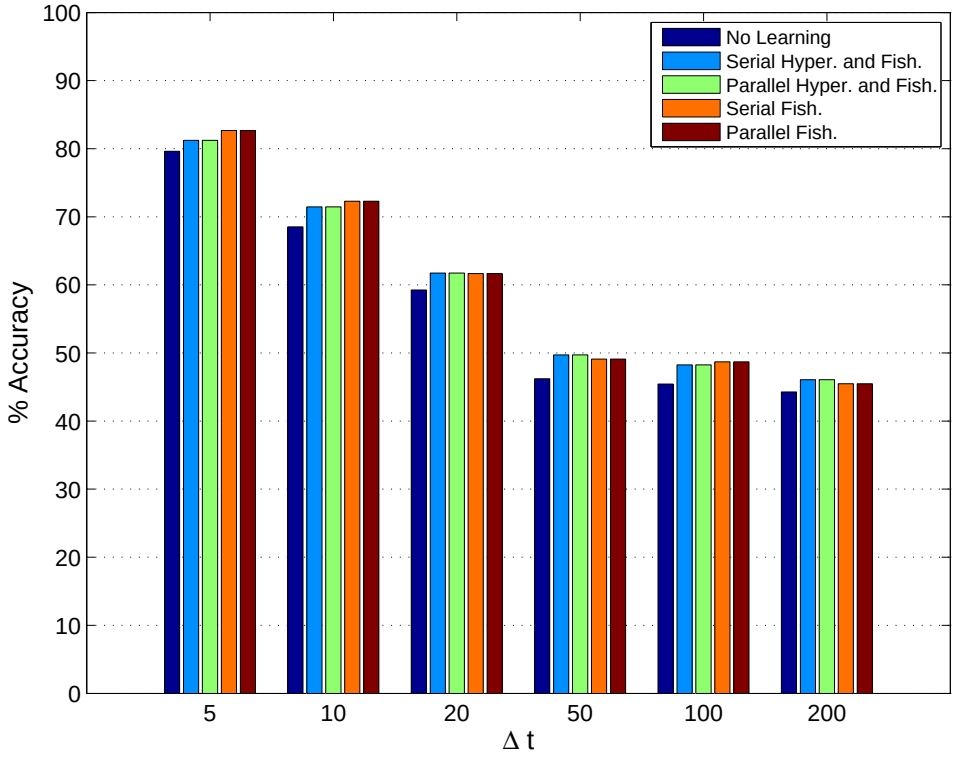
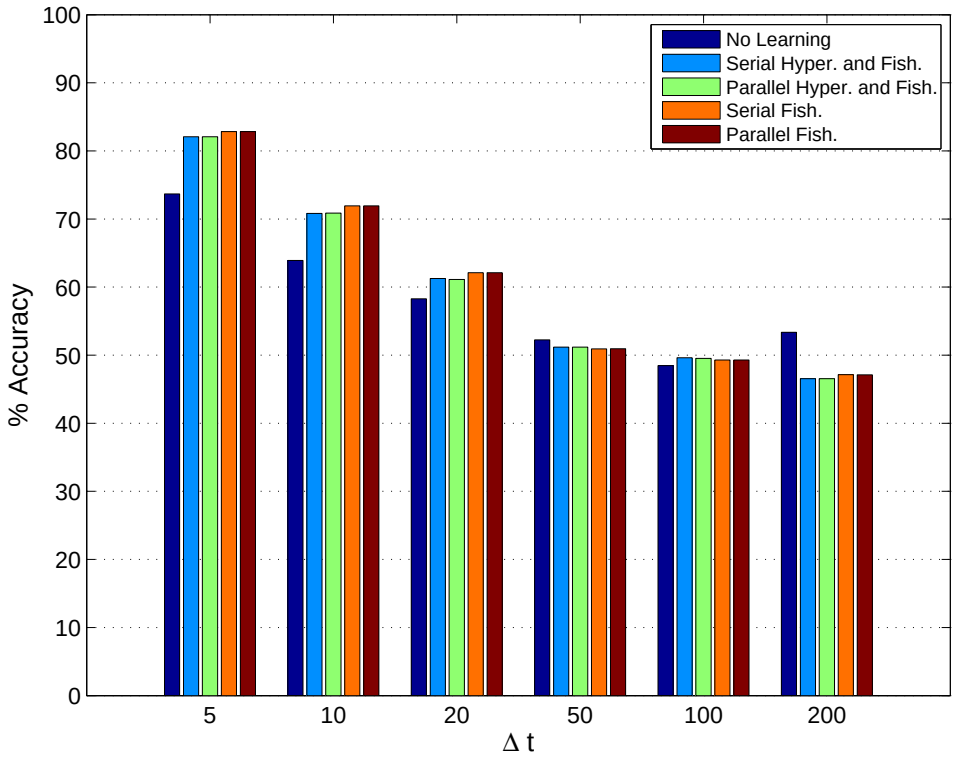


Figure 26: Comparison of % accuracy between kernel learning methods for Subset B



For the full MKL set, Figure 24 shows that the percentage accuracy with any of the kernel learning methods is marginally better than with no learning at all. For shorter time horizons, only learning the Fisher feature parameters results in higher percentage accuracies than when hyperparameter values are learnt as well. There is very little difference between the accuracies of using parallel and serial optimisation techniques. Looking at Figures 25 and 26, the same can be said for Subsets A and B, other than that not learning the kernel outperforms all the learning methods for $\Delta t = 200$ in Subset B.

Figures 27, 28 and 29 (Tables 32, 33 and 34 in the appendix) compare the percentage reduction in β resulting from kernel learning for the full MKL set, Subset A and Subset B.

Figure 27: β reduction from Kernel Learning for full MKL set

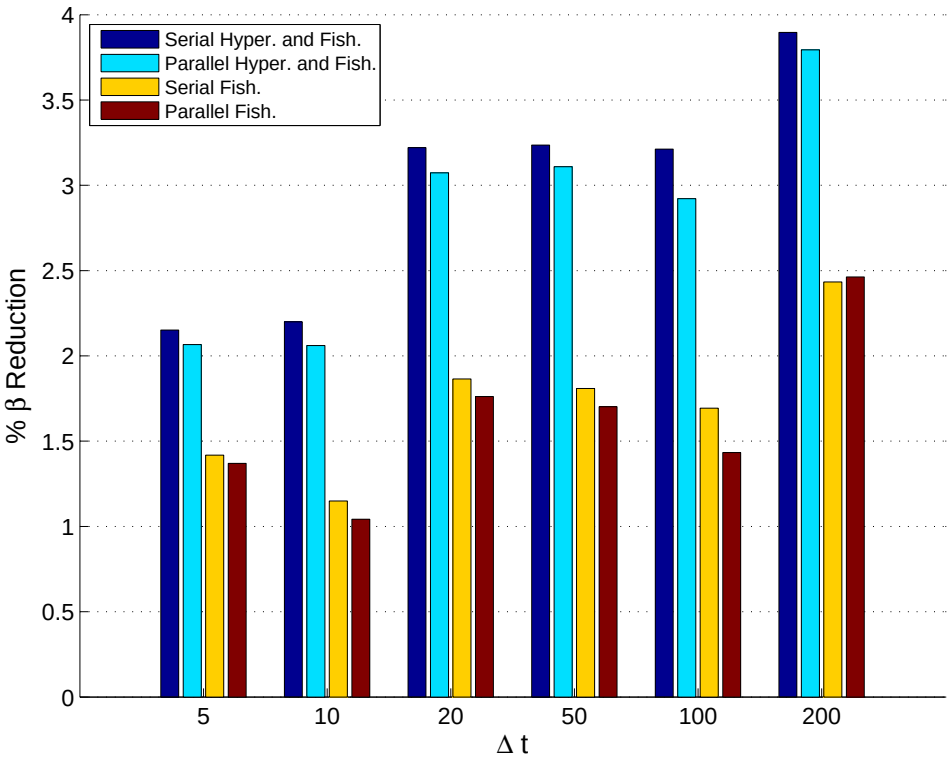


Figure 28: % β reduction from Kernel Learning for Subset A

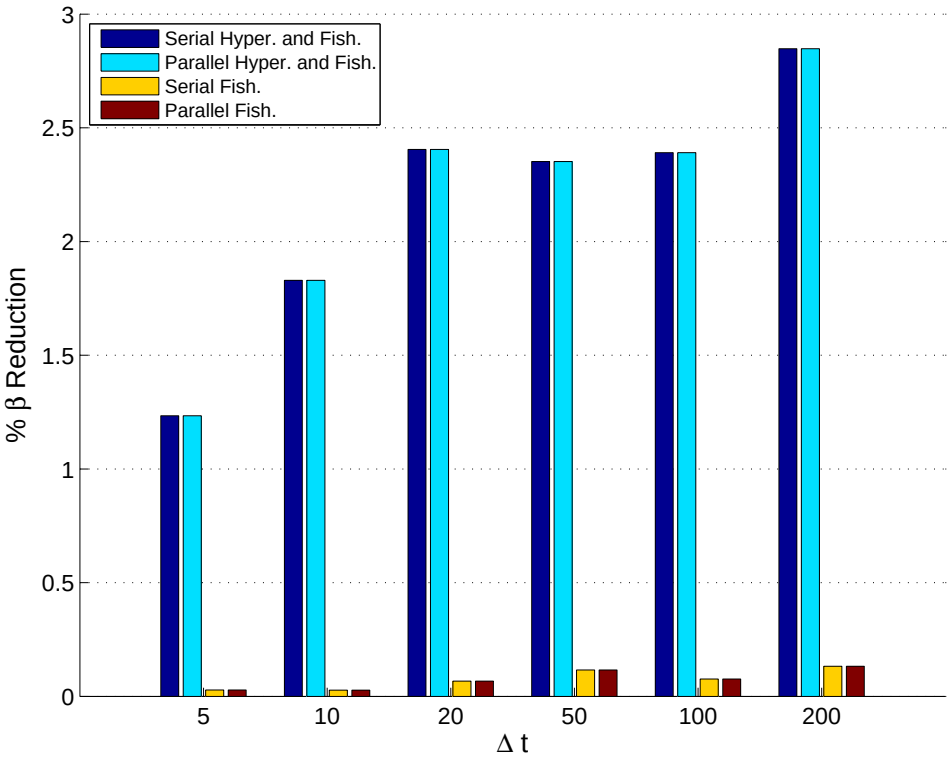
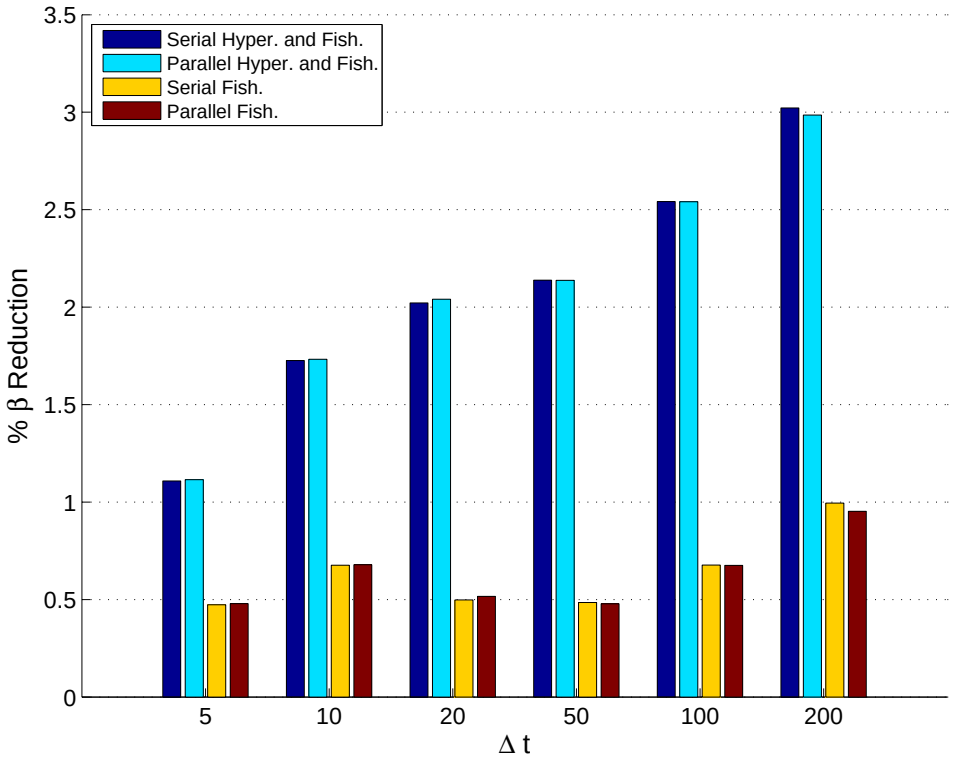


Figure 29: % β reduction from Kernel Learning for Subset B



Once can see in Figure 27 that the percentage reduction in β is more significant for the full MKL set when kernel learning optimises hyperparameter values as well as Fisher features, instead of the latter in isolation. This effect is very significant for Subsets A and B, as shown in Figures 28 and 29. In all cases, there is little difference in the percentage reduction between parallel and serial optimisation methods. Furthermore, the reduction is more pronounced for longer time horizons. It is worth noting that although Subset A does not contain any Fisher features, and hence would not benefit from kernel learning directly, the process outlined in Section 9.1 still allows a reduction in β through the MKL stage - this is why there is a non-zero percentage reduction in β shown for the serial and parallel Fisher feature parameter optimisation methods in Figure 28.

Figures 42, 43 and 44 in the appendix show scatter-plots of the relationship between percentage reduction in β and the resulting percentage accuracy of the predictor for the full MKL set, Subset A and Subset B. The corresponding correlation coefficients for these relationships are shown in Figure 30.

Figure 30: % correlation between % β reduction and % accuracy

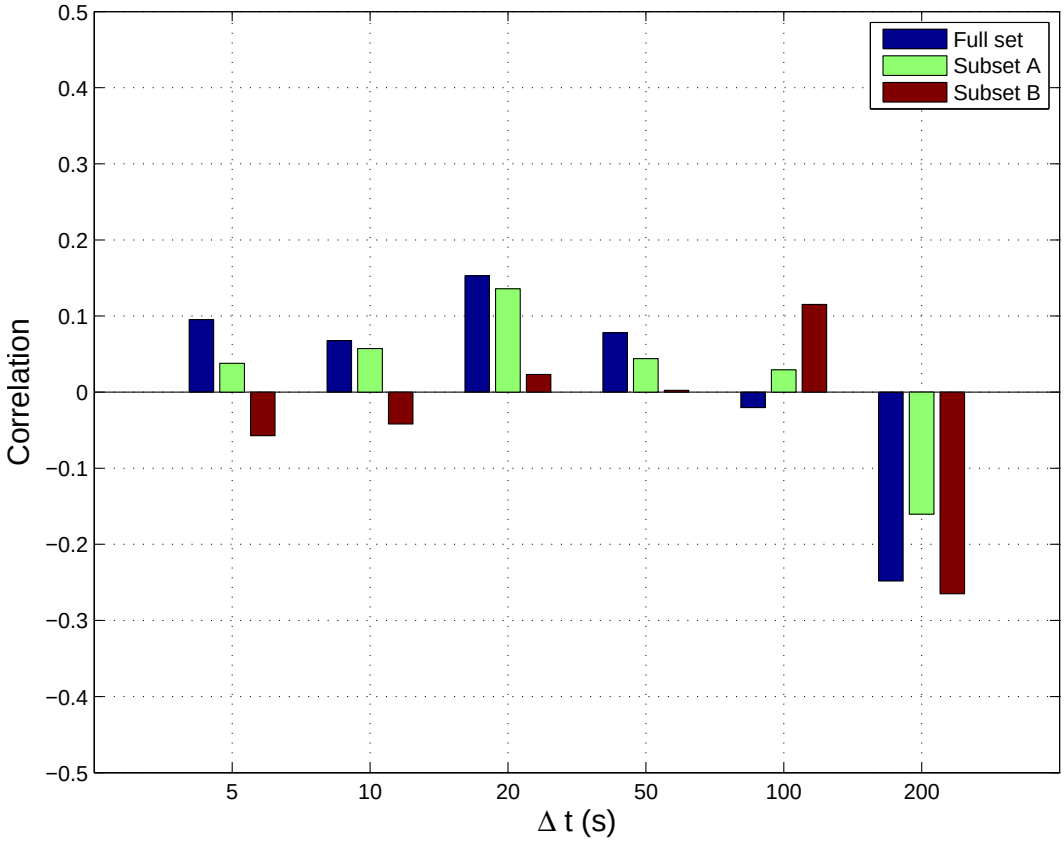


Figure 30 (Table 35 in the appendix) shows that the correlation between percentage β reduction and resulting percentage accuracy of the predictor is generally weakly positive for all the learning methods with the exception of the longest predictive horizon where it is moderately negative.

Figures 42, 43 and 44 in the appendix show scatter-plots of the relationship between final β value (i.e. after kernel learning) and the resulting percentage accuracy of the predictor for the full MKL set, Subset A and Subset B. The corresponding correlation coefficients for these relationships are shown in Figure 31.

Figure 31: % correlation between final β value and % accuracy

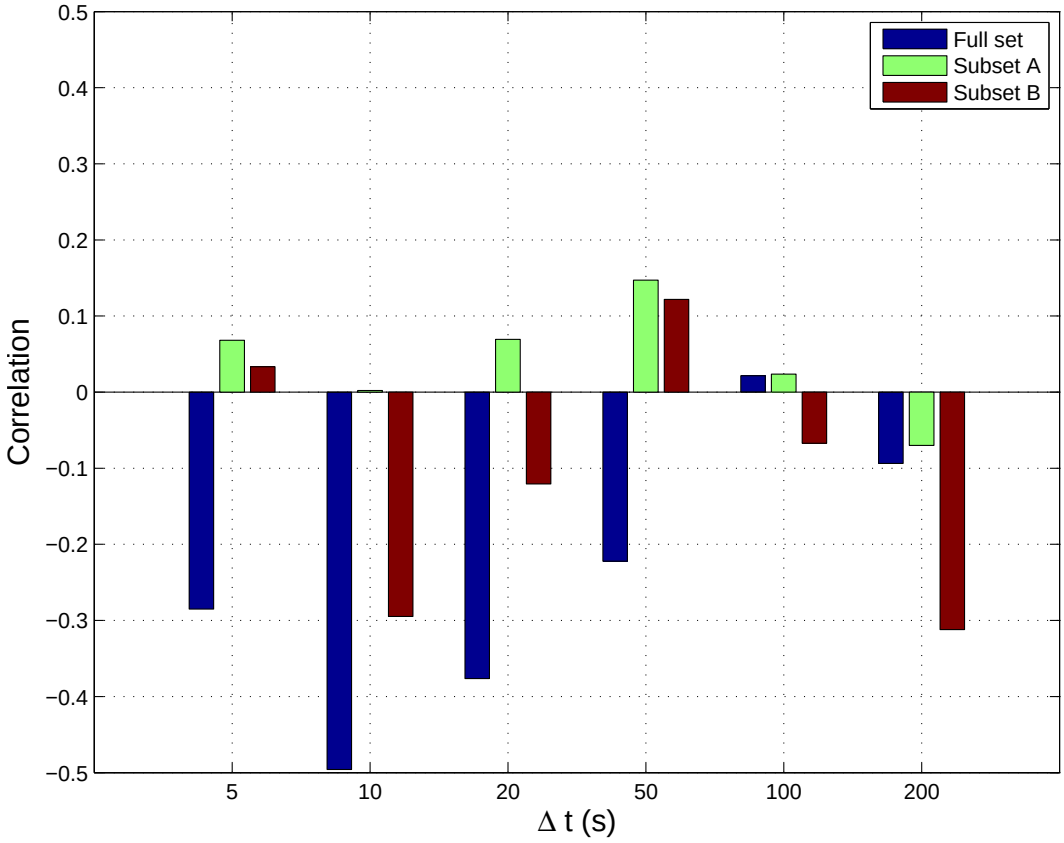
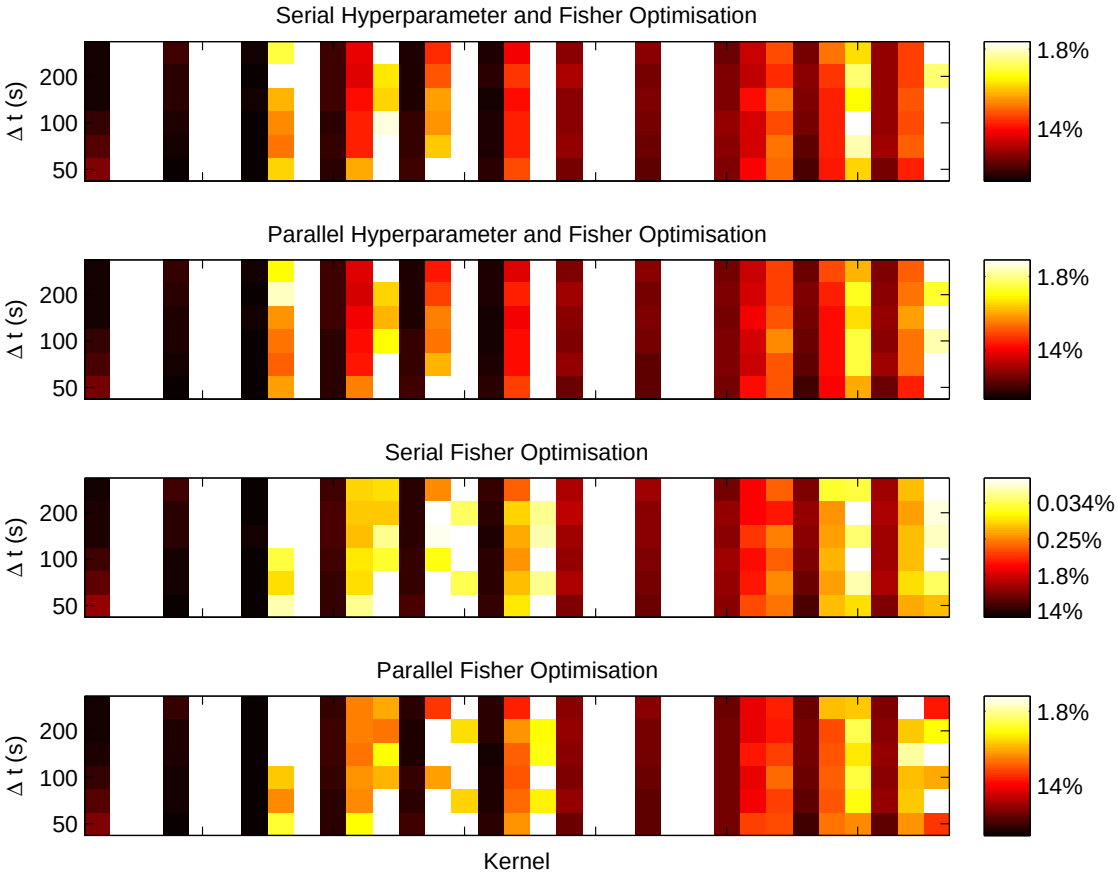


Figure 31 (Table 36 in the appendix) shows that the correlation between percentage β reduction resulting percentage accuracy of the predictor is generally negative for all the learning methods.

11.3.2 Weightings

Figures 32, 33 and 34 compare the average weightings allocated by MKL using the four different kernel learning methods for the full MKL set, Subset A and Subset B.

Figure 32: Comparison of average weighting across kernel learning methods for full MKL set



The first observation that one can make regarding Figure 32 is that there is strong consistency in the weightings that the kernels are allocated across the learning methods. It is also evident that many of the kernels are awarded no weighting at all and only kernels 1, 4, 7, 10, 13 and 16 (F_1RBF, F_2RBF, F_3RBF, F_4RBF, F_5RBF, F_6RBF) are awarded more than 10% on average. There is also strong consistency across the different time horizons.

Figure 33: Comparison of average weighting across kernel learning methods for Subset A

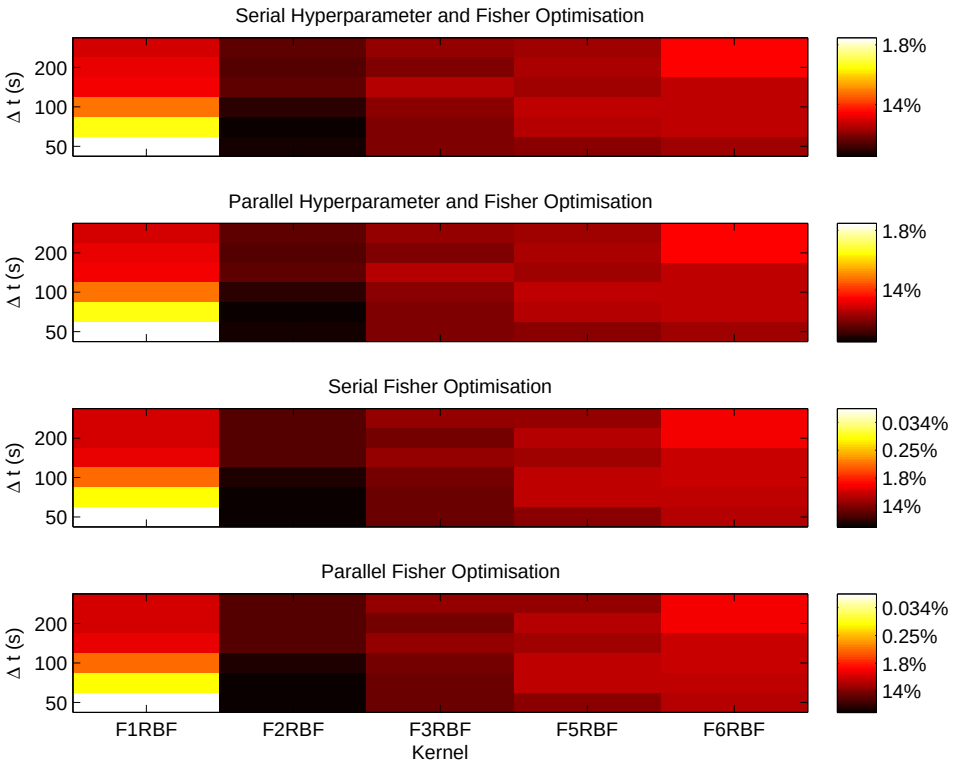
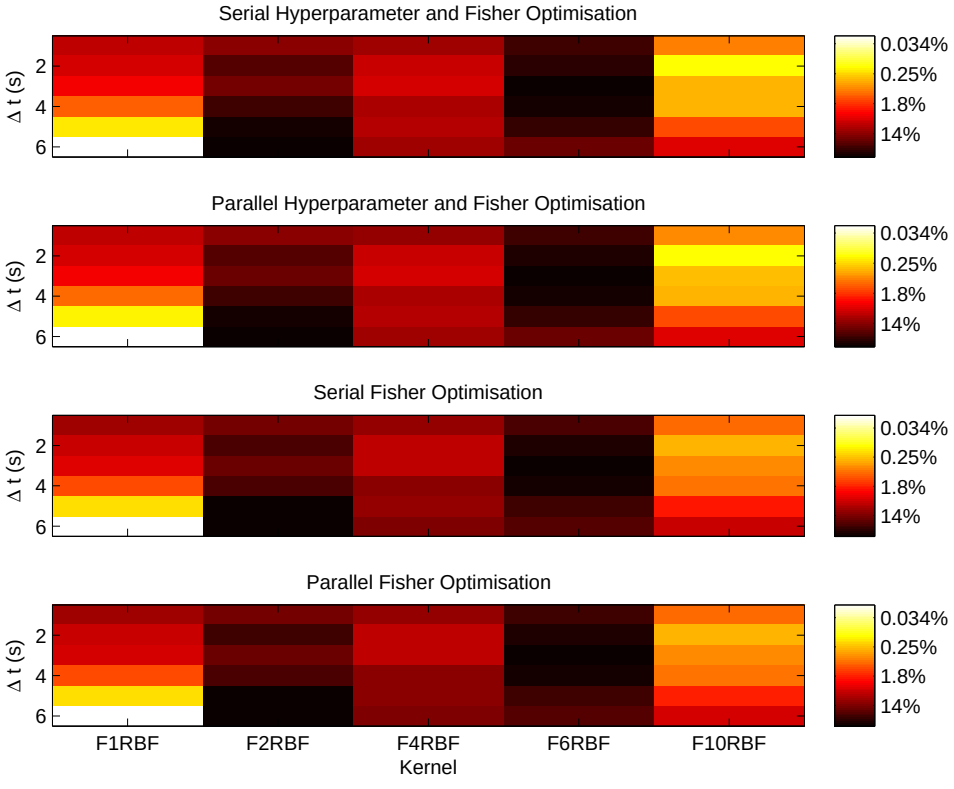


Figure 34: Comparison of average weighting across kernel learning methods for Subset B



The consistency across learning methods and time horizons is also noticeable for Subsets A and B through Figures 33 and 34, the former only allocating F_2RBF average weightings $> 10\%$ and the latter allocating F_2RBF and $F_6RBF > 10\%$ weighting on average. *Refer to Section 11.2.1 for a reminder of what each of the subset kernels are.*

11.3.3 Parameter values

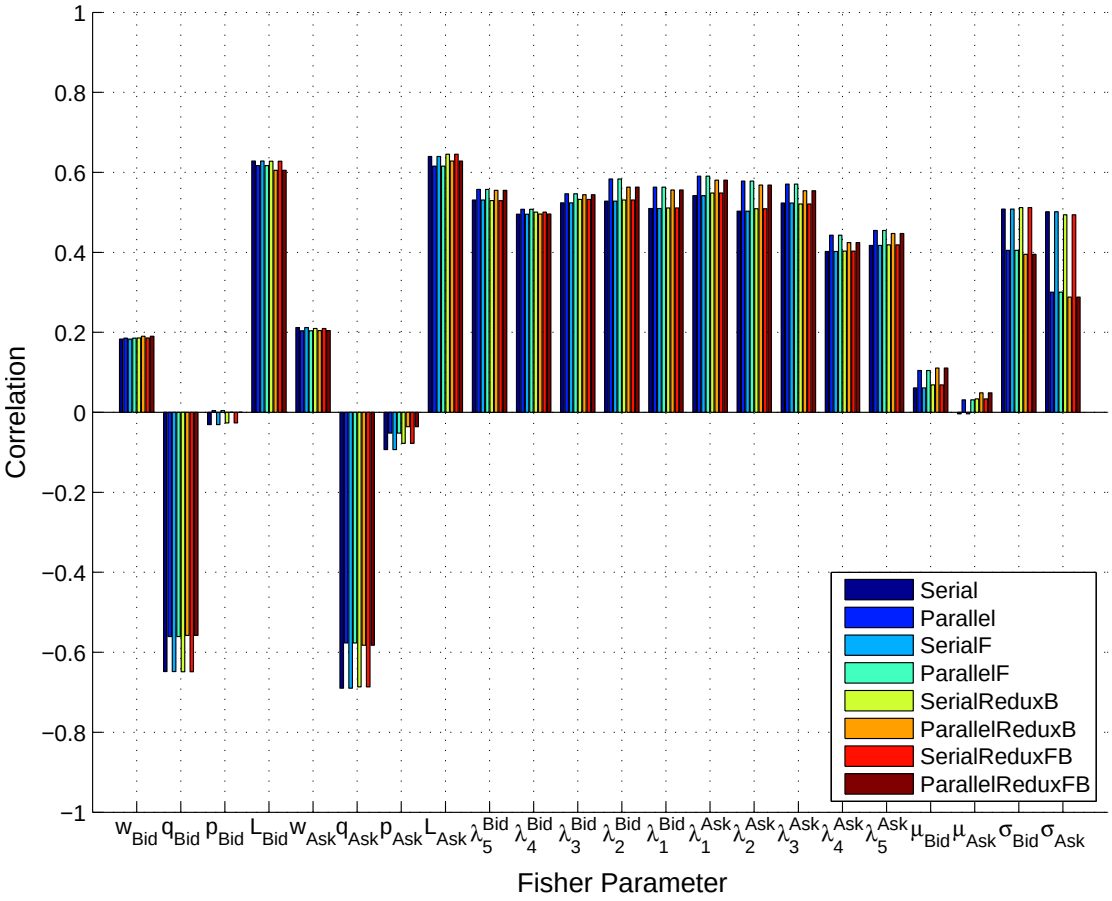
For the sake of completeness, Figures 48 to 59 in the appendix show how the hyperparameter values learnt in kernel learning for the full MKL set change over time for the three different mapping types (RBF, Poly and INN) and four different optimisation methods (serial hyperparameter and Fisher optimisation, parallel hyperparameter and Fisher optimisation, serial Fisher optimisation and Parallel Fisher optimisation).

Figures 60 to 70 in the appendix show the three Fisher models' parameters that were learnt using the four optimisation methods (serial hyperparameter and Fisher optimisation, parallel hyperparameter and Fisher optimisation, serial Fisher optimisation and Parallel Fisher optimisation) for the full MKL set and Subset B over time. These figures show for the majority of the parameters that the time series of the learnt parameter value in question is very similar across the different sets and across the distinction between whether optimisation is altering the hyperparameters as well as the Fisher parameters. However, the time series of parameter values learnt using parallel learning is typically much more stable / of lower variance than using a serial optimisation method.

Figures 71 to 81 in the appendix compare the Fisher models' parameters that were learnt against their Maximum Likelihood estimates over time. In the majority of cases, though the time series of parameter values is clearly related, the learnt values exhibit much higher variance than the Maximum Likelihood estimates of the relevant parameter value.

Figure 35 (Table 37 in the appendix) shows the correlation between these learnt and ML estimates of the Fisher parameters for the different optimisation methods.

Figure 35: Correlation between learnt and ML estimates of Fisher parameters



One can see in Figure 35 that the correlation between the ML parameters and those derived from kernel learning is significantly positive in the majority of cases. The exceptions being parameters w_{Bid} & w_{Ask} from the ACD model and μ_{Bid} & μ_{Ask} from the Wiener model where it is weakly positive, p_{Bid} & p_{Ask} from the ACD model where it is weakly negative and q_{Bid} & q_{Ask} from the ACD model where it is strongly negative. The results are consistent across the different MKL sets and learning methods, with the correlations always stronger for techniques involving serial optimisation than those using parallel optimisation.

11.4 Practical considerations

In order to investigate the computational complexity of each technique, the time taken to run each technique's corresponding experiment was recorded. These computation times are shown in Figure 36.

Figure 36: Computational time of experiments

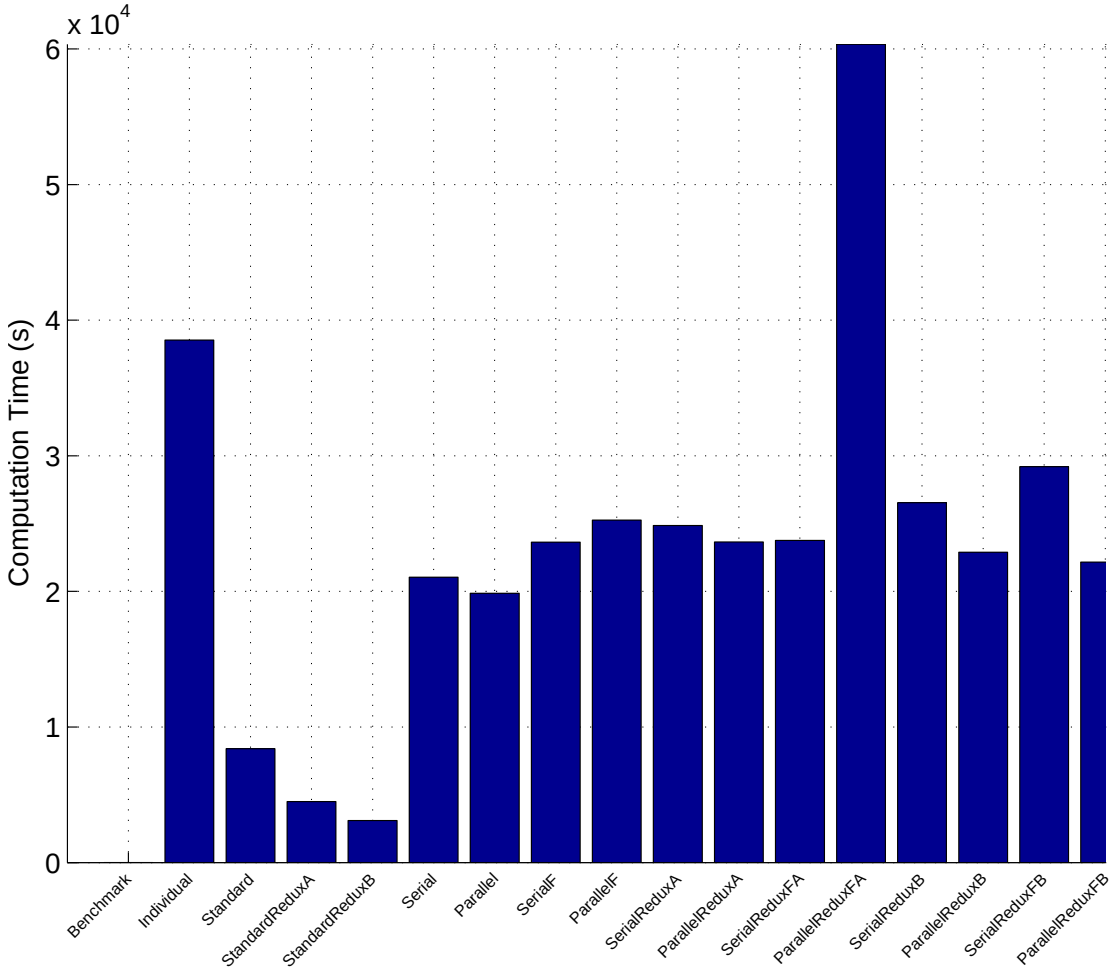


Figure 36 (Table 38 in the appendix) shows that the kernel learning methods are more computationally expensive than their standard MKL counterparts. The experiments involving kernel learning are fairly consistent in their computational times, with the exception of ParralelReduxFA, which takes over twice as long to complete as its counterparts. Running the SVM individually takes more time than any of the methods other than ParralelReduxFA. The benchmark's computation time is so low that it is indistinguishable from zero when shown on the same scale as that of the other methods, reflecting its highly simplistic construction

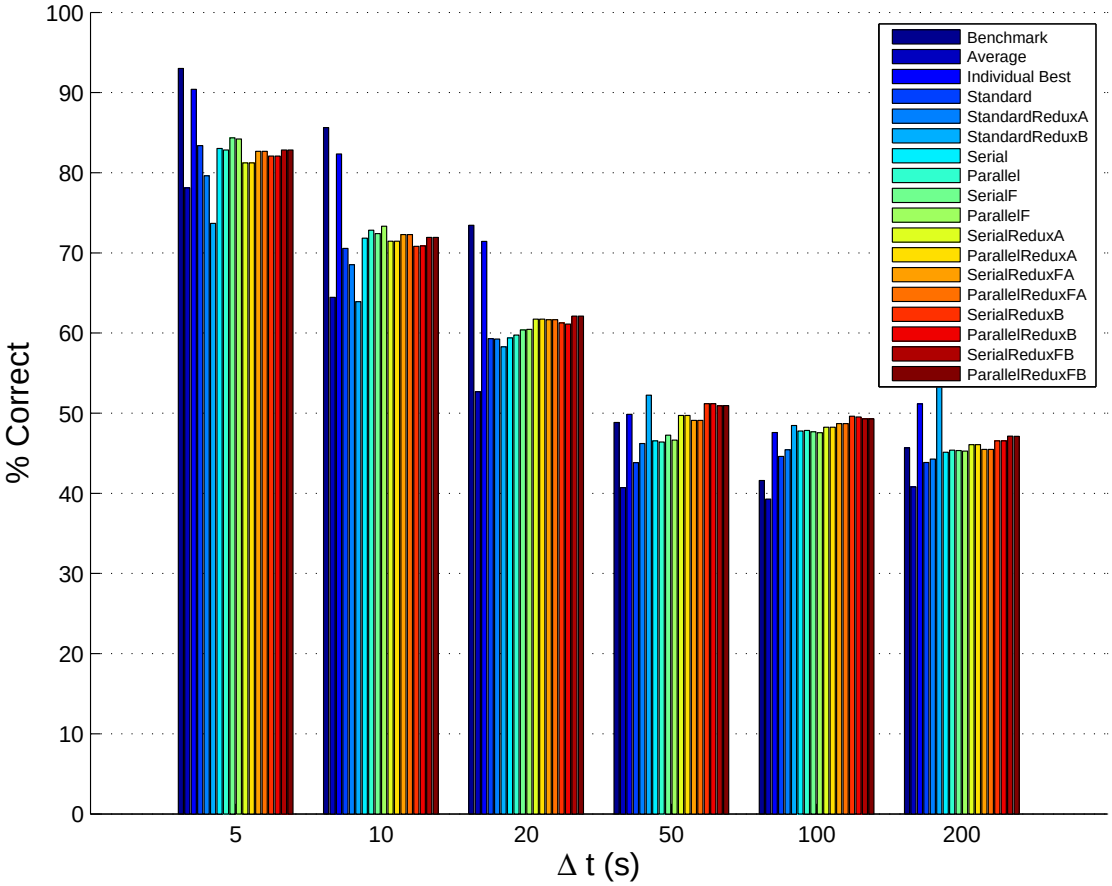
In order to be able to compare the MKL methods with the set of individual SVM in terms of percentage accuracy and frequency of predictions, the individual SVM with the highest percentage accuracy for each time horizon were selected and referred to as if they were one individual SVM. This aggregate kernel *Individual Best* was composed as follows:

Table 11: SVM that *Individual Best* is composed from

Δt	**Kernel**
5	F_6RBF_1
10	$F_{10}RBF_1$
20	F_2RBF_1
50	F_2RBF_1
100	F_1RBF_2
200	F_1RBF_5

Figure 37 (Table 39 in the appendix) shows an overall accuracy comparison between the different methods.

Figure 37: Overall experimental accuracy comparison



One can see from Figure 37 that for shorter time horizons ($\Delta t \leq 20$) the benchmark outperforms all the other methods, while the methods based on Subset B have the highest percentage accuracy for the longer time horizons. The average kernel has generally the lowest performance across all time horizons, while the best individual kernel is reasonably similar to the MKL methods for the longer time horizons, but significantly higher for the shorter ones.

Figure 38 (Table 40 in the appendix) shows the percentage number of times predictions were possible for all the different methods.

Figure 38: Proportion of times predictions possible for different methods

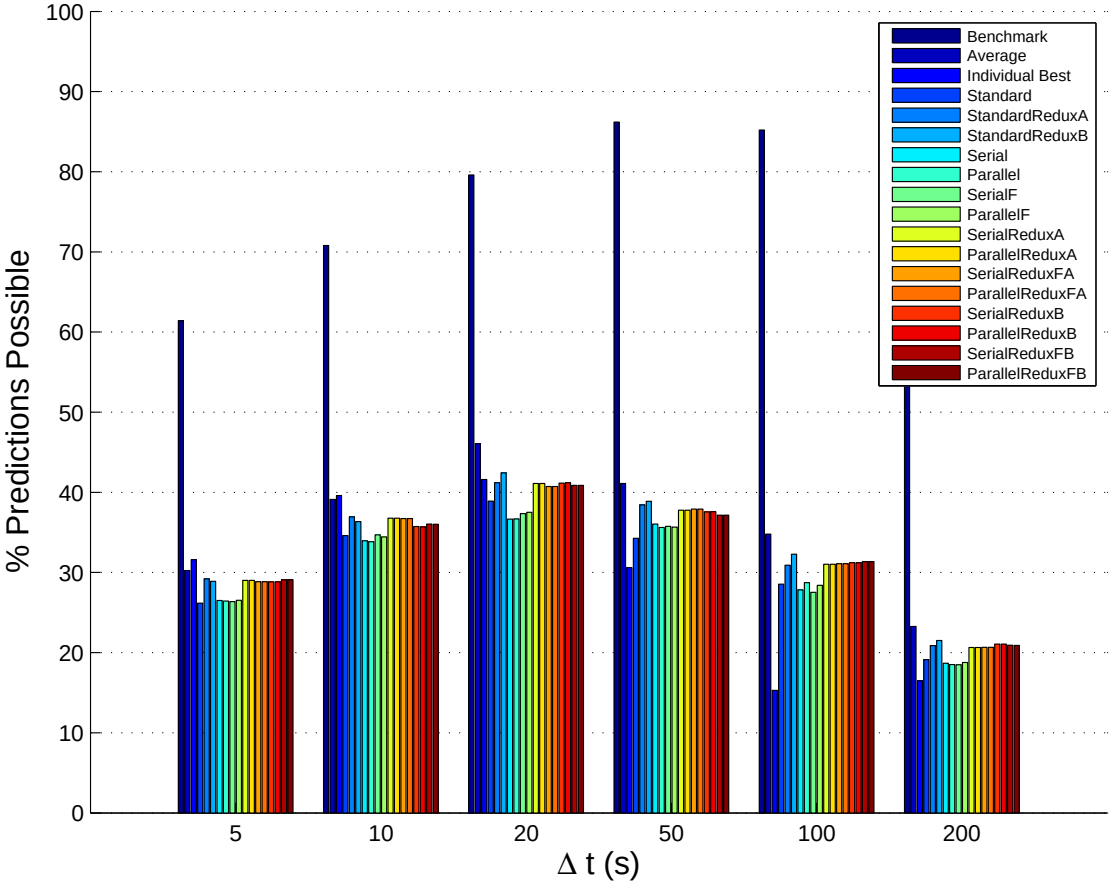


Figure 38 shows that the benchmark is able to make predictions significantly more frequently than any of the other methods. Following this, the average kernel is able to make predictions more frequently than any of the other techniques. Less noticeably, methods based on Subsets A and B are able to make predictions slightly more frequently than those based on the full MKL set. All methods are able to make predictions most frequently over medium time horizons (i.e. $20 \leq \Delta t \leq 50$), with the number of predictions possible falling as the time horizon is increased or decreased.

12 Experimental conclusions

12.1 General kernel selection

12.1.1 Accuracy comparison

Referring to Figure 8 in Section 11.1.1 one can clearly see that the benchmark has a very high percentage accuracy for short time horizons, exceeding that of all the individual SVM for $\Delta t \leq 20$s. If one considers the benchmark's construction, this significant accuracy over shorter time periods is not surprising; one would expect trends that are evident in the 100s of in-sample training data to persist for a reasonable time with the effect tailing off with the longer forecast horizons over which the trend is less likely to have persisted to. The fact that the accuracies of the individual SVM are so different to the benchmark for the shorter time horizons highlights the fact that they are doing something different to simply trend following, and for longer time horizons, actually something more effective.

One can see from Figure 9 in Section 11.1.1 that there is no feature that is the most effective across all time horizons and that it is not possible to generalise on the features' performance across the different predictive time horizons. However, grouping the features into different classes (price, volume or Fisher feature based), as shown in Figure 11, shows much greater consistency. For all time horizons $\Delta t \leq 100s$, the volume-based features show the highest predictive accuracy with the Fisher features following closely behind. The price based features only show relative outpeformance for the longest time horizon $\Delta t = 200s$. The relative outperformance of volume based features to price based ones for shorter time horizons (which is reversed for longer time horizons) is exactly what one would expect to see when market microstructural effects, related to order book volumes and the features based on them, have a much stronger effect on future price movements than previous price action. It seems that in the case of EURUSD, the cut-off in effectiveness of volume based features vs price based ones occurs somewhere between 100 and 200s. The Fisher features' performance is only marginally better than price based features for the majority of time horizons, indicating that it might be making predictions in a very similar manner to the price based features.

Another interesting observation from Figure 9 is that the performance of the individual kernels is more differentiated for shorter time horizons than for longer ones,

where the percentage accuracy of each individual SVM is much more similar to each other.

In terms of the effectiveness of the different kernel mappings highlighted in Figure 10, the RBF clearly significantly outperforms the Polynomial and Infinite Neural Network mappings across all time horizons. The performance of the the latter being very similar in their mediocrity. All the literature has pointed to the RBF being the most suitable kernel mapping for financial prediction tasks, and the results clearly back this up.

Something that is so obvious that it could be overlooked is the general decrease in accuracy of the predictive methods that occurs for all the different ways of generalising across feature type, feature class and kernel mapping. Accuracy generally declines to $100s$ and then slightly improves at $200s$. One would expect accuracy to decline with time; the trend-following component that all the predictors have inherent in their design (through having a bias factor in the SVMs they are based on) means that a proportion of their predictive ability can always be ascribed to trend-following and this component's effectiveness will decline with increased time horizon. The greatest increase in predictive accuracy when moving from $\Delta t = 100s$ to $200s$ is shown when generalising across feature class in the price feature. This is also seen in Figure 9 with the significant increase in performance of features F_1 and F_2 when moving from $\Delta t = 100s$ to $200s$. This indicates that the price based features class, and in particular the two moving averages of price based features F_1 and F_2, work well for shorter time horizons and longer ones, but not for the intermediate horizon of $100s$.

Though individual kernels are able to outperform the benchmark for longer time horizons, in no cases does generalising (i.e. averaging) over the categories of feature, class or kernel mapping outperform the benchmark. In other words, for individual SVM the results show that no particular feature, feature class or kernel mapping outperforms the benchmark for any time horizon.

12.1.2 Stability comparison

Referring to Figure 12 in Section 11.1.2, given the benchmark's trend-following nature one would expect it to be highly consistent in its accuracy for shorter time horizons - trends are unlikely to suddenly reverse in the short term if they have

already persisted for the duration of the in-sample period. However, this is not true for longer time horizons, where a trend has had a much longer time and hence greater chance to change direction. The most effective individual SVM's performance is massively inconsistent for the shorter time horizons. For these shorter time horizons, this SVM does not go through time periods of higher predictive accuracy followed by lower ones, which one would expect from a predictor which was somehow dependent on market regimes or the noisiness of the time series it is attempting to forecast - it is constantly inconsistent. However, for the longest time horizon ($\Delta t = 200s$) the predictor's performance does seem to be marginally less erratic - going through longer periods of time where it is consistently effective or ineffective. This all suggests that the individual SVM is not really doing much more than picking up on some short term trend-following behaviour when employed on the shorter time horizons, but that there is predictive ability over and above this trend-following capacity when the SVM is employed for $\Delta t = 200s$. This is backed up by some of the individual SVMs' overall outperformance in predictive accuracy over the benchmark as shown in Figure 8.

One can see the stabilities shown in Figure 12 quantified in Figure 13, which shows again the benchmark being significantly more stable than the individual SVM for shorter time horizons but much less so for longer ones, with the stability of the individual SVM doing the opposite.

Revisiting what is written in the results concerning stability summarised by feature, kernel mapping and feature class:

- F_{11}, the Wiener model-based Fisher feature, generally has the highest stability out of the features for all time horizons.
- For $\Delta t \leq 20$, the INN is the kernel mapping with the highest stability, followed very closely by the Polynomial kernel mapping. This is switched round for the longer time horizons, but in all cases the RBF is the least stable predictor.
- The feature classes are fairly similar in their stability at all time horizons other than $\Delta t = 200$, where the Fisher feature is significantly more stable than the other two classes.

In summary, the trend-following benchmark is both much more accurate and stable over shorter predictive horizons, whilst some of the individual SVM are more

effective on both counts for longer horizons. It is not really possible to generalise the individual SVM's accuracy across feature or feature class but the RBF kernel mapping outperforms the Polynomial and Infinite Neural Network mappings for all time horizons. This lack of generalisation is not the case when it comes to stability, where F_{11}, the Wiener model based Fisher feature is generally the most consistent predictor.

12.2 Standard MKL

Referring to Figure 16 in Section 11.2, one can clearly see a strong relationship between both how many times an individual kernel was selected by the Simple MKL algorithm and what weightings the individual kernels were allocated and the corresponding accuracy of the predictor when it was used on its own individually. This shows that SimpleMKL is generally both selecting the more effective of the individual SVM for its predictive performance and allocating them more significant weightings. The fact that the correlation is higher for the number of times an individual SVM is selected than the weighting it is allocated indicates that SimpleMKL might be more effective at choosing whether to place an individual kernel in the set it is using for prediction than allocating weightings to the kernels it has selected. The decrease in correlation with predictive time horizon could be due to the fact that there is more to differentiate kernels in their performance, and hence make it easier for Simple MKL to choose between them, for shorter time horizons than for longer ones (as can be seen for example in Figure 9).

SimpleMKL clearly favours features F_1 to F_6 (all the price based features and half of the volume-based features) and the RBF mapping with much higher weightings across all time horizons - as can be seen in Figures 17, 18 and 19. The choice of the RBF kernel mapping can be explained through its significant outperformance in predictive accuracy, but the bias towards features F_1 to F_6 can be neither explained through their predictive accuracy nor their stability. Furthermore, it is hard to quantify a comparison between the consistency in the weightings across different time horizons as shown in Figures 17, 18 and 19 and the consistency between the accuracy or stability of the individual SVM when generalised over feature, feature class or kernel mapping. However, it does appear that the SimpleMKL weightings are much more stable across time than either the accuracy or stability of the individual SVM.

12.2.1 Kernel subset

The kernel subsets described in Section 11.2.1 consist mostly of price-based features in both cases, with the addition of a couple of volume-based features in subset A (where the kernels were selected based on their weighting in the full-set SimpleMKL) and one volume based feature and one Fisher feature in subset B (where the kernels were selected based on the performance as individual SVM).

Subset A's composition reflects Simple MKL's proclivity for price based features, whilst subset B's reflects the fact that a price-based feature often had the highest predictive accuracy for any given time horizon (despite the fact that price based features as a whole did not). The kernel mapping is always RBF in both subsets, reflecting Simple MKL's strong bias to selecting it as a mapping and its significant predictive outperformance relative to the other kernel mappings.

Looking at Figure 20, one can see that reducing the size of the set from which SimpleMKL chooses its kernels to focus on ones that one would suspect would improve performance, either because full-set SimpleMKL allocated them higher weightings or because they had higher predictive accuracies when used individually, does increase performance for the longer time horizons. The fact that it does not do so for the shorter ones suggests again that a large proportion of the predictive ability of the SVM based approach, be it individual SVM or in combination through SimpleMKL, is based on short term trend-following and that the larger set used in full-set MKL is better able to capitalise on this than smaller sets based around more potentially effective predictors. The fact that the benchmark outperforms the other methods for shorter time horizons, but that this is reversed for the longer ones, lends greater credence to this idea. The same is also true for Subset A's shorter time horizon outperformance / longer time under performance relative to Subset B; it is possible that the kernels that Subset B is composed of, being selected because of their outperformance as individual SVM, are doing more genuine prediction over and above the bias component of trend-following than their counterparts in Subset A.

The outperformance of all three MKL methods relative to the average kernel at all time horizons indicates that the weighting procedure that SimpleMKL employs is much more effective when it comes to the resulting predictor than simply allocating all the individual kernels in the full set an equal weighting. There are clearly some individual kernels in this full set that significantly reduce the performance of any predictor that includes them and SimpleMKL is able to weed these out.

Comparing the stability of the different sets, as shown in Figure 21, one can see that they are fairly similar, with the full MKL set being generally the most stable in its performance. One would expect the stability of SimpleMKL predictions to increase slightly with set size - there being occasions when kernels that are typ-

ically not selected by SimpleMKL are suddenly required, owing to some unusual feature of the in-sample data. As mentioned before, given the benchmark's trend-following nature, one would expect it to be highly consistent in its accuracy for shorter time horizons; trends are unlikely suddenly to reverse in the short term if they have already persisted for the duration of the in-sample period (100s), but this is not the case for longer time horizons. The average kernel's high stability relative to the other methods for longer time horizons is likely to be due to the stability inherent in simply averaging the predictions of a large set of predictors as opposed to weighting a smaller subset of predictors with weightings derived from some in-sample data.

The majority of Subset A's predictive accuracy is due to kernel F_2RBF_1, which Figure 22 shows to have the lion's share of the weightings allocated by SimpleMKL across all time horizons. One cannot attribute Subset B's performance so unilaterally, with the weightings being more evenly distributed amongst several kernels, most notably F_2RBF_1, F_1RBF_2 and F_1RBF_5. It is clear that features F_1 and F_2, the two moving average of price based features, are allocated the most significant weightings by SimpleMKL for both subsets.

12.3 Kernel learning

12.3.1 Reduction in β

Figures 24, 25 and 26 in Section 11.3.1 show that for all three MKL methods (full-set, Subset A and Subset B), learning the kernels' hyperparameters / Fisher feature parameters generally represents a slight improvement in terms of percentage accuracy relative to no kernel learning. For shorter time horizons, only learning the Fisher feature parameters results in higher percentage accuracies than when hyperparameter values are learnt as well, though this effect is even more marginal.

There is very little difference between the accuracies of using parallel and serial optimisation techniques. Given the crude iterative nature of the serial optimisation method, the optimisation problem of minimising β as described in Section 9.1 is clearly trivial enough to not warrant parallel optimisation.

The kernel learning's limited improvement in performance is reflected in the low proportional reduction in β that the learning process achieves, as shown in Figures 27, 28 and 29. As one would expect, in all cases the proportional reduction in β is more significant when optimising the hyperparameter values along with the Fisher feature values. The extent of the difference in β reduction between optimising the Fisher feature parameters and doing this as well as optimising the hyperparameter values is due to the exponential effect the hyperparameters have on kernel mappings (e.g. σ in the RBF mapping: $\exp\left(-\left\|\boldsymbol{x}-\boldsymbol{x}'\right\|^2/\sigma^2\right)$) - and hence exert a much stronger influence on β than the Fisher feature parameters. This effect is compounded in Subset B (Figure 29), where only one of the kernels is composed of a Fisher feature. Note that the percentage β reduction is more or less zero for Subset A (Figure 28) when hyperparameter optimisation is not taking place - reflecting the fact that Subset A contains no Fisher feature based kernels to optimise.

Examining the relationship between reduction in β and resulting accuracy more closely, once can see from Figure 30 that the correlation between percentage β reduction and percentage accuracy of the resulting predictor is generally weakly positive. This means that there is a slight relationship between the extent of kernel learning that has taken place and the resulting accuracy. This is not the case for the longest time horizon, where the correlation is negative, indicating that there may be over-learning (i.e. overfitting) taking place for $\Delta t = 200s$.

The stronger negative correlation between the final β value and resulting accuracy, as shown in Figure 31, is an indication that a lower β value is associated with a more effective predictor, and that it is a sensible thing to be attempting to minimise in our iterative SimpleMKL / kernel learning procedure (as described in Section 9.1) from an empirical, as well as a theoretical, perspective.

12.3.2 Weightings

Looking at the weightings awarded to kernels when kernel learning is applied on the full MKL set, as shown in Figure 32, one can see strong consistency in the weightings that the kernels are allocated across the learning methods. It is also evident that many of the kernels are awarded no weighting at all and only kernels F_1RBF, F_2RBF, F_3RBF, F_4RBF, F_5RBF, F_6RBF are awarded more than 10% on average. This is a clear reflection of what was mentioned previously in Section 12.2, with SimpleMKL clearly favouring features F_1 to F_6 (all the price based features and half of the volume based features) and the RBF mapping. This favouritism is still evident despite kernel learning now also taking place.

Kernel learning has not altered SimpleMKL's preferences for any of the subsets' kernel weightings either, with Subset A still only allocating F_2RBF average weightings $> 10\%$ (Figure 33) and Subset B allocating F_2RBF and $F_6RBF > 10\%$ weighting on average (Figure 34).

12.3.3 Parameter values

Referring to Figures 60 to 70 in the appendix, the similarity of the learnt Fisher parameters across the different sets and across the distinction between whether optimisation is altering the hyperparameters as well as the Fisher parameters, is no surprise given the similarity in predictive accuracy between these different methods. Although there is little to distinguish the serial and parallel optimisation methods from a performance perspective, it seems that that the latter is significantly more stable in the learnt values it produces. One would expect this as a consequence of the parallel optimisation method's ability to adjust all the parameter values simultaneously as opposed to iteratively in the manner that serial optimisation is carried out.

As can be seen in Figures 71 to 81 in the appendix, it is encouraging that in the majority of cases the time series of Maximum Likelihood derived Fisher parameters is similar to those that are derived as a consequence of learning the kernel. The fact that the two very different processes of kernel learning through minimising β (as described in Section 9.1) and calculating Maximum Likelihood estimates of the parameters result in similar values, indicates that both processes represent effective ways of establishing the Fisher parameter values and that the generative models that embody these parameters have some plausibility. One potential explanation for the optimised values being so similar to the ML ones is that they are initialised with the mean value of each Fisher parameter over the period investigated. However, as one can see from Figures 71 to 81, over the period in question the ML parameter values deviate significantly from their means so this can only be a minor consideration. One would expect lower stability in the optimised parameter values, which depend on many aspects of the kernel and other external factors, relative to the ML estimates which are based on small sets of features of the data.

It is interesting that this relationship does not hold for parameters q_{Bid} & q_{Ask} from the ACD model, with the two different ways of arriving at these parameter values showing strongly negatively correlated results. This is something that should be investigated in further work. The similarity of correlation between ML and kernel learning across set and learning method (Figure 35) reflects the similarities in percentage accuracy across these distinctions. The fact that the serial optimisation methods exhibit stronger correlation with the ML derived parameters than parallel optimisation ones could be due to the parallel optimisation method having a propensity to converge more rapidly to local minima than the iterative serial optimisation method. However, given the similarity in performance of these two techniques, this is unlikely to be a significant effect.

12.4 Practical considerations

Looking at Figure 36 in Section 11.4, one can see that kernel learning involves significantly more computation time than standard MKL. There is almost no difference in computation time between the serial and parallel optimisation methods, tying up once again with the similarity in their performance. Running the full set of SVM individually takes four times as long as their combined use in standard SimpleMKL, highlighting the computational efficiency of the SimpleMKL method.

When considering the overall percentage accuracy of the 16 different combinations of methods investigated here, it is clear that the benchmark outperforms all the other methods for shorter time horizons (Figure 37). As described before, this is due to the trend-following construction of the benchmark, making it difficult to improve upon over the short term because of the expectation that trends evident in 100s of training data might persist for a reasonable time, with the effect tailing off for longer forecast horizons. For longer time horizons, it is methods based on Subset B that show the highest percentage accuracy. Biasing our set of kernels to choose from in SimpleMKL towards ones that showed strong performance when used individually significantly improves performance relative to standard MKL. Learning the kernel represents an improvement on standard MKL for Subset A for longer time horizons ($\Delta t \geq 50$), but not for Subset B, other than at $\Delta t = 100s$ where it is the strongest of the methods. The outperformance of most of the MKL methods relative to the average kernel at all time horizons indicates that the weighting procedure that SimpleMKL employs is much more effective when it comes to the resulting predictor than simply allocating an equal weighting to all the individual kernels in the full set. There are clearly some individual kernels in this full set that significantly reduce the performance of any predictor that includes them and SimpleMKL is able to weed these out.

As can be seen from Figure 38, the trivial construction of the benchmark enables it to make predictions with much higher frequency than any of the other methods for all time horizons. In a similar fashion, the simple averaging of the full set of individual SVM creates a predictor that is able to forecast more often than its more sophisticated counterparts. There is little to distinguish the other methods from each other in terms of the proportion of times they are able to make predictions, other than the fact that methods based on Subsets A and B are generally able to make predictions more frequently than those based on the full set.

Looking at the computational time, percentage accuracy and proportion of times predictions are possible for each method simultaneously, one can comment on the overall performance of the different methods (*which can be referred to by their experimental names in Figures 36, 37 and 38*):

- The benchmark (*Benchmark*) is the most accurate predictor for shorter time horizons by a significant margin. It is also extremely simple to calculate - requiring almost no computational time relative to the other methods - and is able to make predictions far more frequently than the other methods. For any practical purposes it is clearly the most appropriate method for shorter time horizon prediction (i.e. $\Delta t \leq 20s$).

- Constructing an average kernel (*Average*) from the full set of individual kernels is computationally very expensive (requiring the full set of individual kernels to be calculated) and despite being able to make predictions comparatively frequently, leads to a relatively inaccurate classifier for all time horizons. Out of the methods investigated, it is the least suitable for the prediction task this research is concerned with.

- Using the best individual kernel (*Individual Best*) for each time horizon requires one first to know what this kernel would be; it is therefore computationally expensive, in that one would have to run the full set of individual SVM in order to ascertain this. However, the best individual SVM for each time horizon are only marginally less accurate than the benchmark for the shorter time horizons and though not the most accurate of the techniques for the longer ones, are nevertheless more accurate than standard SimpleMKL. Taking into account how frequently it is possible to make predictions using these individual SVM for the longer time horizons, however, shows that they would be significantly less effective in circumstances where this was an important metric.

- Standard SimpleMKL (*Standard*) (i.e. with no kernel learning and operating on the full set) is a relatively computationally efficient method, resulting in reasonable percentage accuracy and frequency over all the time horizons. It is significantly more effective than the average kernel and individual SVM, but less so than its reduced kernel set and kernel learning counterparts.

- Reducing the size of the sets that SimpleMKL is allowed to select from but still not carrying out kernel learning (*StandardReduxA* and *StandardReduxB*) results in the most computationally efficient methods, with accuracy that is only marginally lower than kernel learning implementations in the majority of cases. In the case of $\Delta t = 200s$ and $\Delta t = 50s$, it is actually the most accurate method - significantly so for the latter. Furthermore, the methods' frequency of predictions are generally the highest of the SimpleMKL methods. The two reduced subset standard MKL methods are the most effective performers overall and would make the best choice of predictive method if computational time represented any form of constraint.

- The methods involving learning the kernel (*Serial, Parallel, SerialF, ParallelF, SerialReduxA, ParallelReduxA, SerialReduxFA, ParallelReduxFA, SerialReduxB, ParallelReduxB, SerialReduxFB* and *ParallelReduxFB*) require significant computational time without reasonable improvements in accuracy over standard MKL for the shorter predictive time horizons but generally less significant improvements for the longer ones. It is only for $\Delta t = 100s$ that kernel learning methods show the highest percentage accuracy (*SerialReduxB* and *ParallelReduxB*). The frequency of predictions is the highest of all the MKL methods when kernel learning is applied on the subsets (*SerialReduxA, ParallelReduxA, SerialReduxFA, ParallelReduxFA, SerialReduxB, ParallelReduxB, SerialReduxFB* and *ParallelReduxFB*). Kernel learning on the subsets therefore represent the best choice of method where the frequency of prediction is a more important consideration than computational time.

13 Market microstructure model insights

An outcome of the experiments is that we are able to come up with estimates for the actual parameter values of which the three market microstructural models are comprised, allowing us to specify more concretely the aspects of the time series each of them aims to describe. Given the similarity of the Maximum Likelihood estimates to those derived through learning the kernel and the relative stability of the former, ML estimates will be used. Furthermore, for the sake of simplicity it will be the means of the relevant parameters' values over time that will be shown.

13.1 ACD model

The ACD model introduced in Section 8.1 is specified:

$$h_t = w + qx_{t-1} + ph_{t-1}$$
$$x_t = h_t\epsilon_t, \quad \epsilon_t \sim Exp(\lambda)$$

where x_t is the duration of the price at time t and h_t is its expected duration. Using the mean ML values for the parameters w, p, q and λ, we have on the Bid side:

$$h_t = 0.24 + 0.33x_{t-1} + 0.16h_{t-1}$$
$$x_t = h_t\epsilon_t, \quad \epsilon_t \sim Exp(0.51)$$

And on the Ask side:

$$h_t = 0.29 + 0.31x_{t-1} + 0.04h_{t-1}$$
$$x_t = h_t\epsilon_t, \quad \epsilon_t \sim Exp(0.46)$$

Referring to Figure 7, the price in our dataset spends about the same amount of time moving upwards as it does downwards, so one would expect the bid and ask prices to have similar durations on average. This fact is reflected in the similarity of the w_{Bid} and w_{Ask} parameters: 0.24 and 0.29 respectively. One can see that the other parameter values are fairly symmetric across the two sides, with the exception of p, which shows greater autocorrelation of the expected duration on the Bid than the Ask on average.

13.2 Poisson process

The Poisson model introduced in Section 8.2 is specified:

$$P(\Delta V_i = x) = \frac{e^{-\lambda_i \tau}(\lambda_i \tau)^x}{x!}$$

where

$$\Delta V_i = \frac{\left|V^i_{t+\tau} - V^i_t\right|}{\tau}$$

and τ is a time interval of 1 second.

Using the mean ML values we have the following table of λ_i:

Table 12: λ values for each depth on the *Bid* and *Ask*

Depth (i)	**Side**	
	Bid	**Ask**
1	11.1	11.9
2	12.5	12.0
3	10.8	14.2
4	10.3	12.3
5	13.3	12.6

Once again, there is reasonable symmetry in the λ values across both sides of the book. It is interesting that the rate of change of volumes is greater away from the top of the book (Depth 1) on both sides. This is fairly typical of most First In First Out (FIFO) order books and is due to the fact that traders leave orders speculatively throughout the book hoping to obtain higher priorities in the order queue, cancelling them if they do not wish them to get executed at that price. Orders at the front of the book would have been cancelled previously whilst that price level was not exposed, so as not to risk getting unwanted order fills. For this reason, it is often the second layer in an order book that shows the most activity in terms of volume changes.

The λ values shown here indicate that there are slightly over $ 10 million of volume changes per second on each of the levels of the EURUSD order book.

13.3 Wiener process

The Wiener process barrier model introduced in Section 8.3 is specified:

$$dp_t = \mu dt + \sigma dz$$

where the price of the asset p_t follows a random walk with drift μ and variance σ^2.

This means that the price movement of an asset over a time period t will be distributed:

$$p_t - p_{t-1} = \Delta p_t \sim N(\mu t, \sigma^2 t)$$

Using the mean ML values for the parameters μ and σ^2 we have on the Bid side:

$$\Delta p_t \sim N(-0.02t, 0.28^2 t)$$

And on the Ask side:

$$\Delta p_t \sim N(0.00t, 0.60^2 t)$$

The fact that the two μ values are so close to zero reflects the negligible overall drift in prices and lack of overall trend of the dataset. The Ask prices appear to have a slightly higher variance than the Bid ones, perhaps indicating that the Ask prices moved around more often whilst the Bid remained constant than vice versa on average.

14 Overall conclusions and contributions

What follows is a summary of all the conclusions that have been made as this research has progressed.

14.1 Literature review

When it comes to making predictions in financial markets, the majority of *published* work in this area is traditionally relatively unsophisticated from a model complexity / mathematically interesting point of view. The econometrics literature concentrates mostly on autocorrelation of returns, whilst the majority of technical analysis is concerned with incorporating previous price action of the time series in question.

More recently, there has been a significant body of work implementing signal processing methods in financial time-series prediction and this has evolved to the usage of some machine learning techniques. However, the repertoire of techniques that has been researched for financial market prediction is very small and falls into the following four areas:

- Neural Networks have been researched heavily. However, the literature indicates that the stochastic nature of their weight initialisation, the fact that they cannot be guaranteed to provide optimal solutions and that they are prone to overfitting have been a severe hindrances to their application in many domains.
- The more recent SVM, which overcome the above major problems, have been a more popular tool in the recent literature. However, one of the main problems of the SVM approach for real-world problems is the selection of the feature-space mapping through the choice of kernel, which is often selected empirically with little theoretical justification.
- The evidence on the efficacy of evolutionary algorithms in financial prediction applications is divided, with the majority of the publications in this area being highly empirical.
- The literature suggests that standard HMM techniques have been effective in very specific applications, for example when predicting daily FX returns.

In terms of the research on order books and the related field of market microstructure, there has been a very significant amount of work done in this area, but it is heavily based on stocks and often relates to characterising features such as liquidity, volatility and spreads, instead of attempting to predict future price action.

14.2 Generative machine learning research

The main weakness of generative machine learning models is that the majority of them, and certainly the two investigated in this research, require a large number of parameters to describe them. Although modelling price action as observations generated from some latent process seems plausible, particularly when incorporating either the known autocorrelation of asset returns or the sudden structural changes that occur in a market, this weakness rendered the SAR-HMM and changepoint models impractical for financial prediction tasks. It is my belief that the significant noise to signal ratio in asset price returns renders the majority of generative models impractical for financial prediction tasks - the large numbers of parameters making it difficult to avoid overfitting.

Furthermore, some of the mathematical assumptions that are used in formulating generative models have no real-life financial motivation. For example, there is no reason that the priors used in the changepoint model described in 4.2.1 should be Gamma and Gaussian distributions, other than to make the maths easy. The choice of probability distributions which the generative models briefly investigated here consist of seems to have no foundation in financial theory and hence represents both a great weakness of both these models and one might expect the majority of other standard generative machine learning methods used to model real-world processes - particularly financial ones.

14.3 FX carry basket prediction

Using a small committee of simple discriminative machine learning techniques to predict FX carry basket returns was highly successful. This was almost certainly mainly due to the predictive value of selecting a set of exogenous features known to have a relationship with the returns being forecasted. In this sense it represents a very good example of how well machine learning techniques can capitalise on sophisticated domain knowledge.

14.4 Comparison of LPBoostMKL and SimpleMKL on order book features

The first set of MKL experiments described in Section 6, showed that the kernels using the radial basis function mapping with the three smallest scale parameters (σ^2) on the simple volumes feature and the change in volumes feature were the most relevant for making predictions over the data set. LPBoostMKL and SimpleMKL selected similar kernels and allocated them very similar weightings, highlighting the consistency between the two techniques.

Section 6 was also the only section where attempts were made to explore the real-world consequences of using the MKL methods for actual trading through their incorporation into a simple trading rule and the P&L that would result. It was found that although the two techniques may make incorrect predictions more often than correct ones, the extent of the moves that work in the techniques' favour more than compensates for the losses of the incorrect ones. This was reflected in the fact that the two MKL techniques clearly outperformed the best individual kernel and the moving average-based trend-following predictor. The profit that the two MKL methods realised in the example shown (Figure 5) was significant both in its magnitude and also in its consistency over time, in that the periods where money was lost (draw-downs) were both infrequent and short-lived.

The final point regarding the comparison of SimlpeMKL to LPBoost is that although they are both solving the same optimisation problem and hence one would expect them to give very similar results (as they do), the main difference between them is that the continuously improving LPBoost algorithm can be stopped at any point prior to convergence to produce a suboptimal classifier; in this sense, one can control the accuracy vs training time trade-off for the method. This aspect of the method is of practical benefit in real-time applications where training time is an important constraint.

The first set of MKL experiments were significant in that no price action or features based on prices were taken into account when predicting future prices - in stark contrast to other research attempting to predict market direction. Aside from the novelty that this contribution represents, this means that any trading rules based on these techniques are likely to complement existing rules well, the majority

of which look at previous price action in some manner or other. Furthermore, the out-performance of the kernel-based techniques for long time horizons over the trend-following benchmark clearly makes them a useful method for locating turning points in time series of EURUSD prices.

14.5 SimpleMKL using financially motivated features

Adding features based on standard concepts from the world of trading to the volume based features, as described in Section 7, changed the kernel combinations most commonly selected by SimpleMKL to those using the RBF mapping with the smallest scale parameter on the exponential moving average crossover feature, the RBF mapping with the largest scale parameter on the price standard deviation / moving average feature and the RBF mapping with the largest scale parameter again on the minimums / maximums feature.

Incorporating these financially motivated features also improved percentage accuracy of the predictions, as highlighted in Table 13.

Table 13: % Accuracy of standard MKL using order book vs financially motivated features

Δt	**Standard Order Book**	**Financially Motivated**
5	95	95
10	90	90
20	81	82
50	66	67
100	51	61
200	45	59

For all the MKL methods there was significant consistency in each of the kernel / feature combinations weightings across the different time horizons - in the majority of cases the weighting for a particular combination is not significantly different between when being used to make a prediction for a short time horizon and a longer term one. Furthermore, the performance of the MKL methods in terms of percentage accuracy was on the whole higher than that of any of the kernel combinations individually.

14.6 Summary of experimental conclusions

- The trend-following benchmark has the highest percentage accuracy and consistency for the shorter time horizons. One would expect trends that are evident in in-sample training data to persist for a reasonable time with the effect tailing off with the longer forecast horizons over which the trend is less likely to have persisted.

- The individual SVM constructed from volume based features show the highest predictive accuracy for $\Delta t \leq 100s$ and the price based ones for $\Delta t = 200s$. This is what one would expect to see when market microstructural effects, related to order book volumes and the features based on them, have a much stronger effect on future price movements than previous price action.

- As expected from the literature, the RBF kernel mapping significantly outperforms the Polynomial and Infinite Neural Network ones.

- SimpleMKL tends to choose the highest performing individual SVM and typically allocates them higher weightings.

- SimpleMKL is more effective at choosing kernels to include than it is at allocating them weightings.

- SimpleMKL favours price based features over volume or Fisher feature based ones.

- For all the MKL methods investigated, there is a great deal of consistency with the weightings allocated across the different predictive horizons.

- Reducing the set of kernels that SimpleMKL can choose from to a smaller (biased) set increases performance for the longer time horizons.

- The average kernel, created by allocating equal weightings to all the kernels in the full set, has very poor performance.

- Learning the kernels' hyperparameters / Fisher feature parameters generally improves percentage accuracy relative to no kernel learning.

- There is very little difference between the accuracies of using parallel and serial optimisation techniques to learn the kernel.

- The proportional reduction in β that kernel learning achieves is limited.

- There is a weak relationship between the extent of kernel learning that has taken place and the resulting accuracy of the predictor, indicating that kernel learning is not resulting in over-fitting in the majority of cases.
- There is a reasonably strong relationship between the final β value of a predictor and its accuracy - a lower value being associated with a more effective predictor.
- In the majority of cases, the time series of Maximum Likelihood derived Fisher parameters is similar to those derived as a consequence of learning the kernel.
- The reduced subset MKL methods without kernel learning are the most effective performers overall and would make the best choice of predictive method if computational time represented any form of constraint.
- The methods involving learning the kernel require significant computational time with reasonable improvements in accuracy over standard MKL for the shorter predictive time horizons, but generally less significant improvements for the longer ones. They represent the best choice of method where the frequency of prediction is an important consideration but computational time is less so.

14.7 Summary of financial insights

The first insight in this area from the experiments conducted is the strength of the simple trend-following benchmark. This shows that for shorter time horizons in FX, trends are much more likely to persist than not and that as a consequence the best prediction for EURUSD's movement is what it has been doing in the recent past. The predictive performance of such a naive trend-following technique reduces considerably with increased predictive time horizons to the point where it is more or less ineffective when a predictive time horizon of several minutes is concerned. This conclusion is in contrast to the literature review, which suggested that short term FX returns were highly negatively autocorrelated - see for example Dacorogna *et al.* (2001) [17].

Another useful insight is that order book volumes are more useful than previous price action for making predictions over the short term (i.e. $< 100s$) and vice versa for the longer term.

14.8 Summary of market microstructure model insights

Estimates for the actual parameter values that comprise the three market microstructural models were made, permitting us to specify more concretely the aspects of the time series each of them aims to describe.

It was found that there was a great deal of symmetry between the Bid and Ask parameters for the ACD model, with the exception of p which showed greater autocorrelation of the expected duration on the Bid than the Ask on average.

There was reasonable symmetry in the λ values across both sides of the book for the Poisson model, with the rate of change of volumes being greater away from the top of the book (Depth 1) on both sides.

The two μ parameters for the Wiener process model were close to zero, reflecting the negligible overall drift in prices and lack of overall trend of the dataset. The Ask prices appeared to have a slightly higher variance than the Bid ones, perhaps indicating that on average the Ask prices moved around more often whilst the Bid prices remained constant than vice versa.

15 Potential criticisms and suggestions for further work

The primary criticism of the research this thesis describes is that although machine learning techniques have been shown to be effective in making predictions regarding financial time series in a variety of contexts, the relative outperformance of these techniques has not been extensive and only limited effort has been made in comparing the techniques' performance to benchmarks composed of traditional methods as highlighted in the Literature Review section, particularly in the areas of econometrics and technical analysis. Further research could involve comparing each of the techniques investigated here against more sophisticated financial forecasting techniques than the simple trend-following methods used here.

When the research was started, generative models seemed like a plausible method for financial market prediction due to their representation of these time series as significantly noisy observations of an underlying signal or latent state transition. However, because of what was felt were intrinsic weaknesses of generative models in general and because of the poor results of the experiments that were conducted in this area, the research into generative models employed on their own was abandoned. There is a gap in the research in that only a very small subset of generative machine learning models were experimented with - namely the SAR HMM and changepoint models - and further work could involve investigating the abundance of other generative models, for example Gaussian Mixture Model HMM, self-transition HMM and resetting HMM models [132] to name but a few. The seeming plausibility of the SAR HMM and changepoint models from a financial perspective make them suitable candidates for Fisher kernels - something further work could also investigate.

Furthermore, a greater effort could have been made in attempting to make the generative methods investigated more successful, perhaps for example by extending the models to incorporate exogenous factors known to have an impact on the time series being predicted.

Another issue that has not been addressed in this research is the usage of distributional predictions. It could have been useful to base trading rules on the confidence one has in a prediction. In SVM, the margin of the classifier acts as a confidence

in the predictions it will produce for out-of-sample data points (see for example [133]), the greater it is the more confidence one would have in future predictions. One could therefore only permit predictions to be made if the classifier's margin was above a certain threshold. In this sense, one could control the trade-off between the frequency of predictions vs their accuracy, which would be of benefit for any proprietary trading rules based on these methods.

As stated at the outset, limited effort has been made in this work to examine the real-world consequence of using the methods investigated here for actual trading. Other than the committee of discriminative techniques used for FX carry basket prediction - which was actually traded in a hedge fund with reasonable success - and the incorporation of the forecasts generated from the methods investigated in the section comparing LPBoostMKL and SimpleMKL on simple order book features into simple trading rules, little effort has been made to investigate how useful these techniques might be when it comes to making money. This is because the focus of this research was intended to be much more theoretical than practical. It was felt that shifting the work to more application-based work would have resulted in a departure from the ambition of this research - namely to marry the machine learning and financial domain - through distractions based on the most suitable trading rule, assumptions regarding transaction costs etc. It would obviously be intellectually and potentially financially rewarding to take the theoretical models developed here and apply them in the real world.

There have been inconsistencies throughout this research on the data sets and consequently the assets investigated. The generative methods concern themselves with futures contracts, the committee of discriminative techniques looks at FX carry basket returns and then the latter, more significant body of work is based on EURUSD order book data. In terms of comparing all the methods researched here, it would have perhaps been better to use similar data-sets throughout. However, this shift was a consequence of the change in focus of direction of my research - on one hand the higher dimensionality of order book data was more suited to SVM-based methods; on the other, it was a result of changes in my career whilst this research progressed. Financial data is difficult and expensive to obtain and I was fortunate enough to obtain the data used throughout as a consequence of my employment and as this changed, so did the data I had access to.

Obvious augmentations to the MKL component of this research include a greater variety of kernel mappings than the three used here, a greater range of price and volume-based features - perhaps incorporating other factors such as time (e.g. rate of change of order book updates) and information from other currencies and further market microstructural models / Fisher features than the three used here. It is possible that increasing the size of the full set of feature / kernel mappings from which MKL was selecting would have improved performance, but it is likely that this improvement would have been small. This is also true of investigating more sophisticated optimisation methods for the kernel learning process over and above the serial and parallel methods used here.

16 Executive summary

In order to briefly summarise the research described here and with a view to closing off the thesis's narrative, I will answer the questions posed in the Introduction that amongst other things this research was designed to address.

Can we make predictions in financial markets using machine learning techniques?

The research shows that machine learning techniques can be theoretically used to make effective predictions in currencies. We have shown that it is possible to predict the direction of movement (up, down or stay within the bid-ask spread) of EURUSD between 5 and 200 seconds into the future with an accuracy ranging from 90% to 53% respectively. We have also shown that it is possible to predict the turning points in the prices of a basket of currencies in a manner which can be exploited profitably (and indeed has been).

How do generative and discriminative machine learning methods compare with each other when it comes to dealing with financial time series?

The majority of generative methods, and certainly the two investigated in this research, require a large number of parameters to describe them. The significant noise to signal ratio in asset price returns hence renders the majority of generative models impractical for financial prediction tasks - the large numbers of parameters making it difficult to avoid overfitting.

Can we improve upon existing machine learning techniques to make them more suitable for financial prediction tasks?

Using Multiple Kernel Learning on financially motivated features was highly successful. However, allowing parameters in the features used, along with the kernels based on them, to be learnt was of very little benefit in terms of predictive accuracy.

What features of financial time series are most useful when making predictions?

Order book volumes are more useful than previous price action for making predictions over the short term (i.e. < 100s) and vice versa for the longer term.

Does incorporating domain knowledge of the financial markets, for example market microstructure, improve our predictive ability?

Using features based on commonly used trading rules significantly improved performance. However, incorporating market microstructural models through Fisher kernels only represented a very minor improvement in predictive accuracy.

17 Appendix

17.1 Market microstructure empirical findings

17.1.1 Limit vs market orders

- There is a grey scale of liquidity supply vs demand in that submitting a market order can be seen as taking liquidity, submitting a limit order far from the Best Bid or Offer (BBO) is providing liquidity and submitting an aggressive limit order lies in-between these extremes.
- One can classify traders placing limit orders as having relatively higher patience for the filling of their trades and hence tolerance of execution risk. Conversely, traders placing market orders are relatively impatient and tolerant of price risk.
- The latter group is often motivated by inventory issues; the more information they have about the true value of the asset, the more liquidity they will take. They are likely to start with limit orders and switch to market orders when they approach deadlines for their inventory requirements.
- These informed traders will often take liquidity at the start of a trading period in an attempt to profit from their information, but as prices move towards the true value they may move to creating liquidity. The opposite is true for the relatively less informed liquidity traders.
- Limit orders are sometimes placed in order to test for the weight of trading in the opposite direction. If a trader realises that the arrival rate of market orders is lower than anticipated, he may cancel an existing order to minimise the execution risk.
- Limit order utility is often a concave function of liquidity demand (i.e. the anticipated frequency of opposing market orders).

17.1.2 Spreads

- Spreads widen with trader impatience, clearing the market for liquidity.
- An increase in trader numbers or decrease in order book depth (total volume in book) will tend to decrease spreads. The latter is because limit orders would be required to queue for longer.
- A finer price lattice has a similar effect to reducing order book depth and hence tends to cause spreads to tighten.

- Spreads tend to increase as the dispersal of trader valuations increases.
- When spreads narrow, there is a tendency for more market orders to be submitted until the rate that they are being submitted at increases to the extent that an equilibrium is reached with slightly wider spreads.
- Spreads tend to widen at the end of the trading day, although in some exchanges they can be their largest for short periods at the beginning of the day.
- An increase in the volatility of the true value leads to an increase in the submission rate of limit orders and also an increased probability of orders being picked off. This in turn leads to wider spreads and then subsequently wider spreads.

17.1.3 Market information / reservation values

- Rapidly taking liquidity by submitting market orders leads to information being imparted as prices converge to their true value.
- Traders typically receive information more quickly than the rate their limit orders are executed.
- Queues of orders may develop on the most attractive prices in a book.
- A big difference between a trader's reservation value and price will mean a greater cost of delayed execution when using a limit order.
- Traders with values close to the price will submit limit orders, but ones with big differences (in either direction) will tend to submit market orders.
- Uncertainty regarding other participants' estimates of the true value leads to spreads widening.
- Mid prices tend to move in the direction of trades, so that for example a buy order would drive the mid price up. There is a greater price impact for trades for less frequently traded assets.
- An increase in order depth at the best price decreases the probability of limit order submissions on the same side, but increases it on the other.

- Adverse selection costs and liquidity supply (as measured through order book depth) are inversely related.

17.2 Generative machine learning derivations

17.2.1 Switching Autoregressive Hidden Markov Models

Forwards recursion (4.3):

$$
\begin{aligned}
p(h_t|v_{1:t}) &= p(h_t|v_t, v_{1:t-1}) \\
&= \frac{p(h_t, v_t|v_{1:t-1})}{p(v_t|v_{1:t-1})} \\
&\propto p(h_t, v_t|v_{1:t-1}) \\
&= \sum_{h_{t-1}} p(h_t, v_t|v_{1:t-1}, h_{t-1}) p(h_{t-1}|v_{1:t-1}) \\
&= \sum_{h_{t-1}} p(v_t|v_{1:t-1}, h_{t-1}, h_t) p(h_t|h_{t-1}, v_{1:t-1}) p(h_{t-1}|v_{1:t-1}) \\
h_t \perp v_{1:t-1}|h_{t-1} \Rightarrow &\sum_{h_{t-1}} p(v_t|v_{1:t-1}, h_{t-1}, h_t) p(h_t|h_{t-1}) p(h_{t-1}|v_{1:t-1}) \\
v_t \perp h_{t-1}|v_{1:t-1}, h_t \Rightarrow &\sum_{h_{t-1}} p(v_t|v_{1:t-1}, h_t) p(h_t|h_{t-1}) p(h_{t-1}|v_{1:t-1}) \\
p(v_t|v_{1:t-1}, h_t) \equiv p(v_t|v_{t-R:t-1}, h_t) & \\
&\Rightarrow \sum_{h_{t-1}} p(v_t|v_{t-R:t-1}, h_t) p(h_t|h_{t-1}) p(h_{t-1}|v_{1:t-1})
\end{aligned}
$$

Backwards recursion (4.4):

$$
\begin{aligned}
p(h_t|v_{1:T}) &= \sum_{h_{t+1}} p(h_t|h_{t+1}, v_{1:T}) p(h_{t+1}|v_{1:T}) \\
h_t \perp v_{t+1:T}|h_{t+1} &\Rightarrow p(h_t|h_{t+1}, v_{1:T}) = p(h_t|h_{t+1}, v_{1:t}) \\
\therefore\ p(h_t|v_{1:T}) &= \sum_{h_{t+1}} p(h_t|h_{t+1}, v_{1:t}) p(h_{t+1}|v_{1:T}) \\
&= \frac{\sum_{h_{t+1}} p(h_{t+1}, h_t|v_{1:t}) p(h_{t+1}|v_{1:T})}{p(h_{t+1}|v_{1:t})} \\
&\propto \sum_{h_{t+1}} p(h_{t+1}, h_t|v_{1:t}) p(h_{t+1}|v_{1:T}) \\
&= \sum_{h_{t+1}} p(h_{t+1}|h_t, v_{1:t}) p(h_t|v_{1:t}) p(h_{t+1}|v_{1:T}) \\
h_{t+1} \perp v_{1:t}|h_t &\Rightarrow \sum_{h_{t+1}} p(h_{t+1}|h_t) p(h_t|v_{1:t}) p(h_{t+1}|v_{1:T})
\end{aligned}
$$

17.2.2 Changepoint models

If we assume that we can compute the the predictive distribution conditional on a given run length r_t, i.e. that we can calculate $P(x_{t+1}|r_t, \mathbf{x}_{1:t})$, we can make a prediction about the next observation $P(x_{t+1}|\mathbf{x}_{1:t})$. We do this by summing over r_t the product of the predictive distribution and the posterior distribution of r_t given the observations so far observed:

$$P(x_{t+1}|\mathbf{x}_{1:t}) = \sum_{r_t} P(x_{t+1}|r_t, \mathbf{x}_{1:t}) P(r_t|\mathbf{x}_{1:t}) \tag{17.1}$$

where the posterior can be expressed as the joint probability distribution of the run length r_t and all the observations so far $\mathbf{x}_{1:t}$ divided by the probability of observing all these observations:

$$\begin{aligned} P(r_t|\mathbf{x}_{1:t}) &= \frac{P(r_t, \mathbf{x}_{1:t})}{P(\mathbf{x}_{1:t})} \\ &= \frac{P(r_t, \mathbf{x}_{1:t})}{\sum_{r_t} P(r_t, \mathbf{x}_{1:t})} \end{aligned} \tag{17.2}$$

This joint distribution can be expressed recursively:

$$\begin{aligned} P(r_t, \mathbf{x}_{1:t}) &= \sum_{r_{t-1}} P(r_t, r_{t-1}, \mathbf{x}_{1:t}) \\ &= \sum_{r_{t-1}} P(r_t, r_{t-1}, \mathbf{x}_{1:t-1}, x_t) \\ &= \sum_{r_{t-1}} P(r_t|x_t, r_{t-1}, \mathbf{x}_{1:t-1}) P(x_t|r_{t-1}, \mathbf{x}_{1:t-1}) P(r_{t-1}, \mathbf{x}_{1:t-1}) \end{aligned}$$

Using the notation $\mathbf{x}_{t-1}^{r_{t-1}}$ to represent the set of observations of x associated with the previous time step's run r_{t-1} (i.e. so that $\left|\mathbf{x}_{t-1}^{r_{t-1}}\right| = r_{t-1}$) means that $P(x_t|r_{t-1}, \mathbf{x}_{1:t-1}) \equiv P(x_t|r_{t-1}, \mathbf{x}_{t-1}^{r_{t-1}})$. Furthermore, $r_t \perp \mathbf{x}_{1:t}|r_{t-1} \Rightarrow$

$$P(r_t, \mathbf{x}_{1:t}) = \sum_{r_{t-1}} P(r_t|r_{t-1}) P(x_t|r_{t-1}, \mathbf{x}_{t-1}^{r_{t-1}}) P(r_{t-1}, \mathbf{x}_{1:t-1}) \tag{17.3}$$

where $P(r_t|r_{t-1})$ is defined in (4.5), $P(r_{t-1}, \mathbf{x}_{1:t-1})$ is the result of (17.3) from the previous recursion and $P(x_t|r_{t-1}, \mathbf{x}_{t-1}^{r_{t-1}})$ is the predictive distribution conditional on a given run length, derived in the next section. In order to make the maths appear less cluttered, when r_{t-1} appears as a super-script, we will denote it by $r_{t'}$.

We will use τ to refer to the inverse of the observation variance, i.e. the precision. Furthermore, instead of using a point estimate for τ we will assume that it is distributed according to a Gamma distribution so that $\tau \sim Gam(\alpha, \beta)$. We can then express the predictive distribution $P(x_t|r_{t-1}, \mathbf{x}_{t-1}^{r_{t'}})$ as follows:

$$\begin{aligned}
P(x_t|r_{t-1}, \mathbf{x}_{t-1}^{r_{t'}}) &= \int_0^\infty P(x_t|\tau^{-1})P(\tau^{-1}|r_{t-1}, \mathbf{x}_{t-1}^{r_{t'}})d\tau \\
&= \int_0^\infty P(x_t|\tau^{-1})P(\tau^{-1}|\mathbf{x}_{t-1}^{r_{t'}})d\tau \\
&= \int_0^\infty P(x_t|\tau^{-1})\frac{P(\mathbf{x}_{t-1}^{r_{t'}}|\tau^{-1})P(\tau^{-1})}{P(\mathbf{x}_{t-1}^{r_{t'}})}d\tau \\
&= \frac{1}{K}\int_0^\infty N(x_t|\mu, \tau^{-1})N(\mathbf{x}_{t-1}^{r_{t'}}|\mu, \tau^{-1})Gam(\tau|\alpha, \beta)d\tau \\
&= \frac{1}{K}\int_0^\infty N(\mathbf{x}_t^{r_{t'}}|\mu, \tau^{-1})Gam(\tau|\alpha, \beta)d\tau \text{ where } \left|\mathbf{x}_t^{r_{t'}}\right| = r_{t-1}+1 \\
&= \frac{1}{K}\int_0^\infty \left(\frac{\tau}{2\pi}\right)^{\frac{r_{t'}+1}{2}} \exp\left\{-\frac{\tau}{2}\sum_{i=0}^{r_{t'}}(x_{t-i}-\mu)^2\right\}\frac{\beta^\alpha \exp^{(-\beta\tau)}\tau^{\alpha-1}}{\Gamma(\alpha)}d\tau \\
&= \frac{1}{K}\frac{\beta^\alpha}{\Gamma(\alpha)}\left(\frac{1}{2\pi}\right)^{\frac{r_{t'}+1}{2}}\int_0^\infty \tau^{\alpha+\frac{r_{t'}}{2}-\frac{1}{2}}\exp\left\{-\tau\left[\sum_{i=0}^{r_{t'}}\frac{(x_{t-i}-\mu)^2}{2}+\beta\right]\right\}d\tau
\end{aligned}$$

Substituting in $z = \tau\Delta$ where $\Delta = \sum_{i=0}^{r_{t'}}\frac{(x_{t-i}-\mu)^2}{2} + \beta$ so that $dz = d\tau\Delta$:

$$\begin{aligned}
P(x_t|r_{t-1}, \mathbf{x}_{t-1}^{r_{t'}}) &= \frac{1}{K}\frac{\beta^\alpha}{\Gamma(\alpha)}\left(\frac{1}{2\pi}\right)^{\frac{r_{t'}+1}{2}}\Delta^{-\alpha-\frac{r_{t'}}{2}-\frac{1}{2}}\int_0^\infty z^{\alpha+\frac{r_{t'}}{2}-\frac{1}{2}}\exp(-z)dz \\
&= \frac{1}{K}\frac{\beta^\alpha}{\Gamma(\alpha)}\left(\frac{1}{2\pi}\right)^{\frac{r_{t'}+1}{2}}\Delta^{-\alpha-\frac{r_{t'}}{2}-\frac{1}{2}}\Gamma(\alpha+\frac{r_{t'}+1}{2}) \\
&= \frac{1}{K}\frac{\Gamma(\alpha+\frac{r_{t'}+1}{2})}{\Gamma(\alpha)}\beta^\alpha\left(\frac{1}{2\pi}\right)^{\frac{r_{t'}+1}{2}}\Delta^{-\alpha-\frac{r_{t'}}{2}-\frac{1}{2}}
\end{aligned}$$

Substituting back in $\Delta = \sum_{i=0}^{r_{t'}} \frac{(x_{t-i}-\mu)^2}{2} + \beta$ and also $\alpha = \frac{\nu}{2}$ and $\beta = \frac{\nu}{2\lambda}$:

$$\begin{aligned}
P(x_t|r_{t-1}, \mathbf{x}_{t-1}^{r_{t'}}) &= \frac{1}{K}\frac{\Gamma\left(\frac{\nu}{2}+\frac{r_{t'}+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)}\left(\frac{\nu}{2\lambda}\right)^{\frac{\nu}{2}}\left(\frac{1}{2\pi}\right)^{\frac{r_{t'}+1}{2}}\left(\frac{\nu}{2\lambda}+\sum_{i=0}^{r_{t'}}\frac{(x_{t-i}-\mu)^2}{2}\right)^{-\frac{\nu}{2}-\frac{r_{t'}+1}{2}} \\
&= \frac{1}{K}\frac{\Gamma\left(\frac{\nu}{2}+\frac{r_{t'}+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)}\left(\frac{\nu}{2\lambda}\right)^{\frac{\nu}{2}}\left(\frac{1}{2\pi}\right)^{\frac{r_{t'}+1}{2}}\left[\frac{\nu}{2\lambda}\left(1+\frac{2\lambda}{\nu}\sum_{i=0}^{r_{t'}}\frac{(x_{t-i}-\mu)^2}{2}\right)\right]^{-\frac{(\nu+r_{t'}+1)}{2}} \\
&= \frac{1}{K}\frac{\Gamma\left(\frac{\nu}{2}+\frac{r_{t'}+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)}\left(\frac{1}{2\pi}\right)^{\frac{r_{t'}+1}{2}}\left(\frac{\nu}{2\lambda}\right)^{-\frac{r_{t'}+1}{2}}\left(1+\frac{\lambda}{\nu}\sum_{i=0}^{r_{t'}}(x_{t-i}-\mu)^2\right)^{-\frac{(\nu+r_{t'}+1)}{2}} \\
&= \frac{1}{K}\frac{\Gamma\left(\frac{\nu}{2}+\frac{r_{t'}+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)}\left(\frac{\lambda}{\nu\pi}\right)^{\frac{r_{t'}+1}{2}}\left(1+\frac{\lambda}{\nu}\sum_{i=0}^{r_{t'}}(x_{t-i}-\mu)^2\right)^{-\frac{(\nu+r_{t'}+1)}{2}}
\end{aligned} \tag{17.4}$$

The constant K is the Marginal Likelihood $P(\mathbf{x}_{t-1}^{r_{t'}})$ and is calculated in a very similar way:

$$\begin{aligned}
P(\mathbf{x}_{t-1}^{r_{t'}}) &= \int_0^\infty P(\mathbf{x}_{t-1}^{r_{t'}}|\tau^{-1})P(\tau^{-1})d\tau \\
&= \int_0^\infty N(\mathbf{x}_{t-1}^{r_{t'}}|\mu,\tau^{-1})Gam(\tau|\alpha,\beta)d\tau \\
&= \int_0^\infty \left(\frac{\tau}{2\pi}\right)^{\frac{r_{t'}}{2}}\exp\left\{-\frac{\tau}{2}\sum_{i=1}^{r_{t'}}(x_{t-i}-\mu)^2\right\}\frac{\beta^\alpha \exp^{(-\beta\tau)}\tau^{\alpha-1}}{\Gamma(\alpha)}d\tau \\
&= \frac{\beta^\alpha}{\Gamma(\alpha)}\left(\frac{1}{2\pi}\right)^{\frac{r_{t'}}{2}}\int_0^\infty \tau^{\alpha+\frac{r_{t'}}{2}-1}\exp\left\{-\tau\left[\sum_{i=1}^{r_{t'}}\frac{(x_{t-i}-\mu)^2}{2}+\beta\right]\right\}d\tau
\end{aligned}$$

Substituting in $z = \tau\Delta$ where $\Delta = \sum_{i=1}^{r_{t'}} \frac{(x_{t-i}-\mu)^2}{2} + \beta$ so that $dz = d\tau\Delta$:

$$\begin{aligned}
P(\mathbf{x}_{t-1}^{r_{t'}}) &= \frac{\beta^\alpha}{\Gamma(\alpha)}\left(\frac{1}{2\pi}\right)^{\frac{r_{t'}}{2}}\Delta^{-\alpha-\frac{r_{t'}}{2}}\int_0^\infty z^{\alpha+\frac{r_{t'}}{2}-1}\exp(-z)dz \\
&= \frac{\beta^\alpha}{\Gamma(\alpha)}\left(\frac{1}{2\pi}\right)^{\frac{r_{t'}}{2}}\Delta^{-\alpha-\frac{r_{t'}}{2}}\Gamma(\alpha+\frac{r_{t'}}{2}) \\
&= \frac{\Gamma(\alpha+\frac{r_{t'}}{2})}{\Gamma(\alpha)}\beta^\alpha\left(\frac{1}{2\pi}\right)^{\frac{r_{t'}}{2}}\Delta^{-\alpha-\frac{r_{t'}}{2}}
\end{aligned}$$

Substituting back in $\Delta = \sum_{i=1}^{r_{t'}} \frac{(x_{t-i}-\mu)^2}{2} + \beta$ and also $\alpha = \frac{\nu}{2}$ and $\beta = \frac{\nu}{2\lambda}$:

$$\begin{aligned}
P(\mathbf{x}_{t-1}^{r_{t'}}) &= \frac{\Gamma\left(\frac{\nu}{2}+\frac{r_{t'}}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)}\left(\frac{\nu}{2\lambda}\right)^{\frac{\nu}{2}}\left(\frac{1}{2\pi}\right)^{\frac{r_{t'}}{2}}\left(\frac{\nu}{2\lambda}+\sum_{i=1}^{r_{t'}}\frac{(x_{t-i}-\mu)^2}{2}\right)^{-\frac{\nu}{2}-\frac{r_{t'}}{2}} \\
&= \frac{\Gamma\left(\frac{\nu}{2}+\frac{r_{t'}}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)}\left(\frac{\nu}{2\lambda}\right)^{\frac{\nu}{2}}\left(\frac{1}{2\pi}\right)^{\frac{r_{t'}}{2}}\left[\frac{\nu}{2\lambda}\left(1+\frac{2\lambda}{\nu}\sum_{i=1}^{r_{t'}}\frac{(x_{t-i}-\mu)^2}{2}\right)\right]^{-\frac{(\nu+r_{t'})}{2}} \\
&= \frac{\Gamma\left(\frac{\nu}{2}+\frac{r_{t'}}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)}\left(\frac{1}{2\pi}\right)^{\frac{r_{t'}}{2}}\left(\frac{\nu}{2\lambda}\right)^{-\frac{r_{t'}}{2}}\left(1+\frac{\lambda}{\nu}\sum_{i=1}^{r_{t'}}(x_{t-i}-\mu)^2\right)^{-\frac{(\nu+r_{t'})}{2}} \\
&= \frac{\Gamma\left(\frac{\nu}{2}+\frac{r_{t'}}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)}\left(\frac{\lambda}{\nu\pi}\right)^{\frac{r_{t'}}{2}}\left(1+\frac{\lambda}{\nu}\sum_{i=1}^{r_{t'}}(x_{t-i}-\mu)^2\right)^{-\frac{(\nu+r_{t'})}{2}} \qquad (17.5)
\end{aligned}$$

Substituting $K = P(\mathbf{x}_{t-1}^{r_{t'}})$ from (2.5) into (2.4) gives the final expression for $P(x_t|r_{t-1}, \mathbf{x}_{t-1}^{r_{t'}})$:

$$P(x_t|r_{t-1}, \mathbf{x}_{t-1}^{r_{t'}}) = \frac{\Gamma\left(\frac{\nu_n}{2}+\frac{1}{2}\right)}{\Gamma\left(\frac{\nu_n}{2}\right)}\frac{1}{\sqrt{\nu_n\pi\sigma_n^2}}\left[1+\frac{1}{\nu_n}\left(\frac{x_t-\mu}{\sigma_n}\right)^2\right]^{\frac{\nu_n+1}{2}} \qquad (17.6)$$

where $\nu_n = \nu + r_{t'}$, $\beta_n = \beta + \frac{1}{2}\sum_{i=1}^{r_{t'}}(x_i-\mu)^2$ and $\sigma_n^2 = \frac{2\beta_n}{\nu_n}$. (17.6) is a Student's t-distribution where the parameters ν and σ are functions of the run length r_{t-1} i.e. $x_t \sim St(\nu^{r_{t'}}, \sigma^{r_{t'}})$.

17.3 Futures contracts

Table 14: List of futures contracts used in SAR HMM and changepoint model research

Contract	Sector
AUDUSD	FX
Canadian 3M	STIR
Soybean Oil	Agg
GBPUSD	FX
Corn	Agg
Crude Oil	Energy
Cotton	Agg
EURUSD	FX
USD Index	FX
Bund	Bond
Bobl	Bond
Schatz	Bond
US 3m	STIR
Dax	Equity
Euribor	STIR
Fed Funds	STIR
5 Year US	Bond
Gold	Comm
Copper	Comm
Heating Oil	Energy
Hang Seng	Equity
Japan 3M	STIR
Japan 10 Year	Bond
JPYUSD	FX
Coffee	Agg
Wheat	Agg
Live Cattle	Agg
Cocoa	Agg
Lean Hogs	Agg
MXPUSD	FX
Nasdaq	Equity
Nat Gas	Energy
Gasoline	Energy
Russell	Equity
Soybeans	Agg
Sugar	Agg
CHFUSD	FX
Nikkei	Equity
Eurostoxx	Equity
2 Year	Bond
10 Year	Bond
30 Year	Bond
AUD 3M	STIR
AUD 10 Year	STIR

17.4 Financial acronyms

This is an explanation of each of the acronyms from Section 5.3.

Max Draw Down: The maximum peak-to-trough decline over the time period, expressed as a percentage of the peak before the decline started.

SD Annualized: This is the average annual standard deviation of all the returns.

CAGR: Compound Annual Growth Rate. This is calculated by taking the n^{th} root of the total percentage growth rate, where n is the number of years being considered.

Max DD Time: The longest peak-to-trough decline over the time period.

Sharpe Ratio: The CAGR divided by the annualised standard deviation of all returns.

Sortino Ratio: The CAGR divided by the annualised standard deviation of the negative returns.

17.5 Price-based features

- $\mathcal{F}_1$: A common trading rule is the moving average crossover technique (see for example [118]) which suggests that the price P_t will move up when its short term moving average EMA_t^{short} crosses above a longer term one EMA_t^{long} and vice versa.

- $\mathcal{F}_2$: Breakout trading rules (see for example [134]) look to see if the price has broken above or below a certain threshold and assume that once the price has broken through this threshold the direction of the price movement will persist. One way of defining this threshold is through the use of Bollinger Bands [135], where the upper/lower thresholds are set by adding/subtracting a certain number of standard deviations of the price movement σ_t^L to the average price MA_L^t for a period L.

- $\mathcal{F}_3$: Another breakout trading rule called the Donchian Trend system [134] determines whether the price has risen above its maximum $\max_t^L$ or below its minimum $\min_t^L$ over a period L and once again assumes that once the price has broken through this threshold the direction of the price movement will persist.

- $\mathcal{F}_4$: The Relative Strength Index trading rule [136] is based on the premise that there is a relationship between the number of times the price has gone up over a period $\Uparrow_t^L$ vs the number of times it has fallen $\Downarrow_t^L$ and assumes that the price is more likely to move upwards if $\Uparrow_t^L > \Downarrow_t^L$ and vice versa.

17.6 Calculation of p-values

- For each in-sample period, the proportion of occurrences of each of the three classes of movement (up, down or none) over the 100 instances of in-sample data was determined.
- Predictions of movement were then generated randomly for each of the instances of the out-of-sample period where a prediction was deemed possible by SimpleMKL / individual kernel, each class having a probability of being assigned based on the in-sample proportions.
- This was repeated 10^5 times for each out-of-sample section with the number of times the randomly generated predictions were correct, along with the number of times SimpleMKL / individual kernel was correct for that period recorded each time.
- The proportion of the 10^5 iterations that the number of correct predictions recorded for all the out-of-sample periods was greater than that reported by SimpleMKL / individual kernel was used to calculate the P-value.
- In the work reported here, not one of the 10^5 iterations of randomly generated predictions outperformed the SimpleMKL / individual kernel methods.

17.7 Further figures

Figure 39: Stability of individual SVM

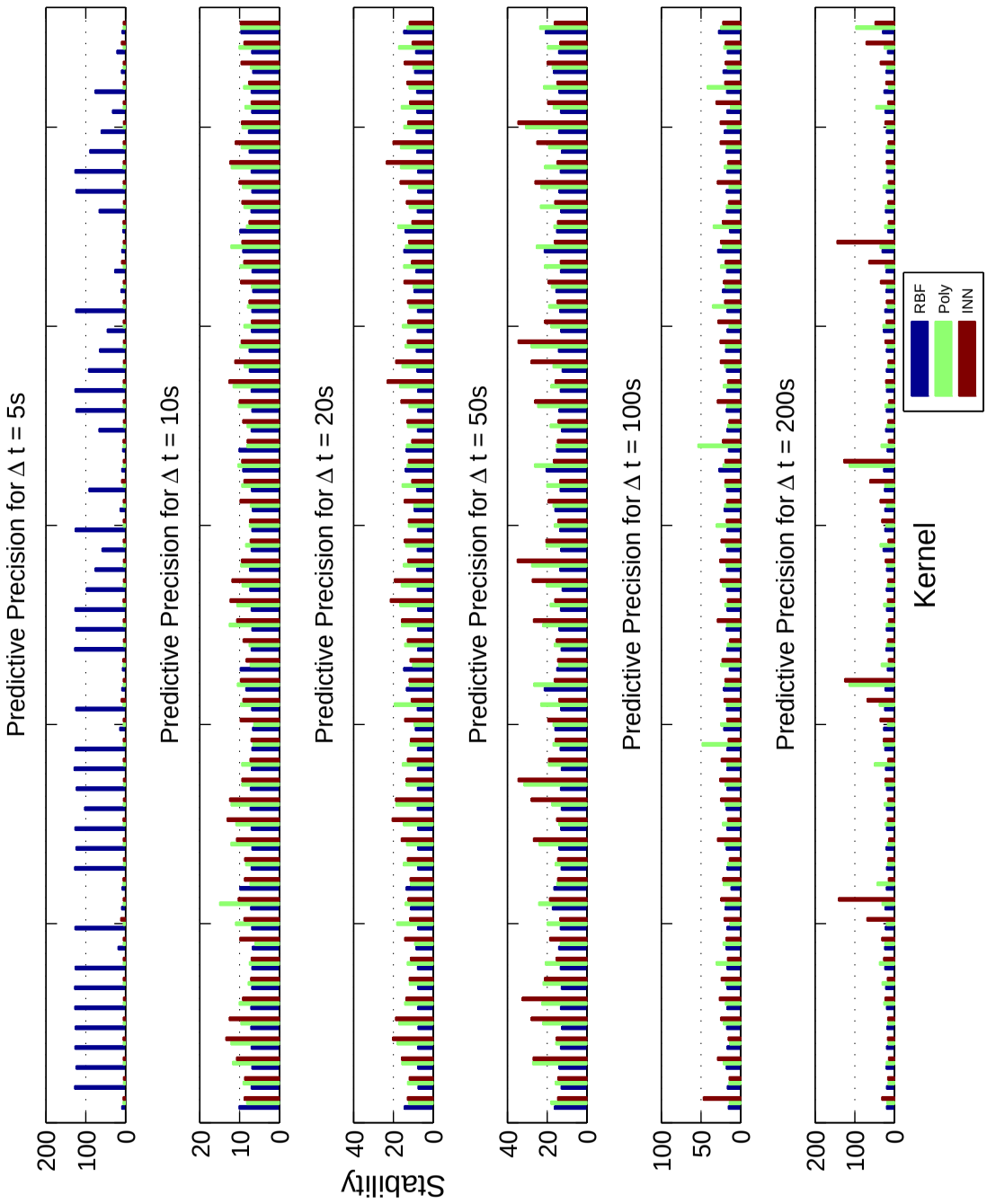


Figure 40: MKL average weighting & number of times included vs % accuracy

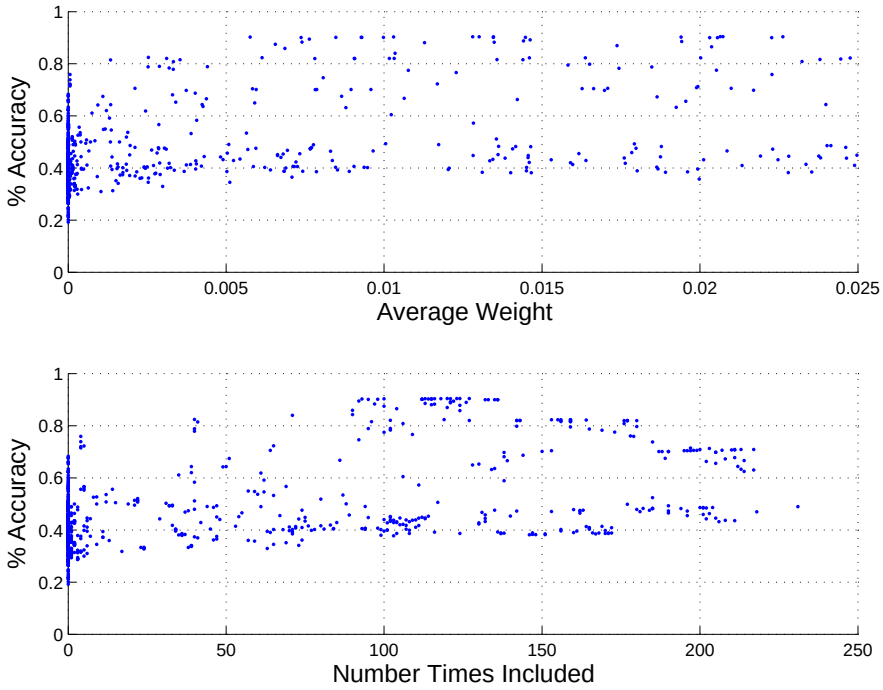


Figure 41: Average weighting of individual SVM

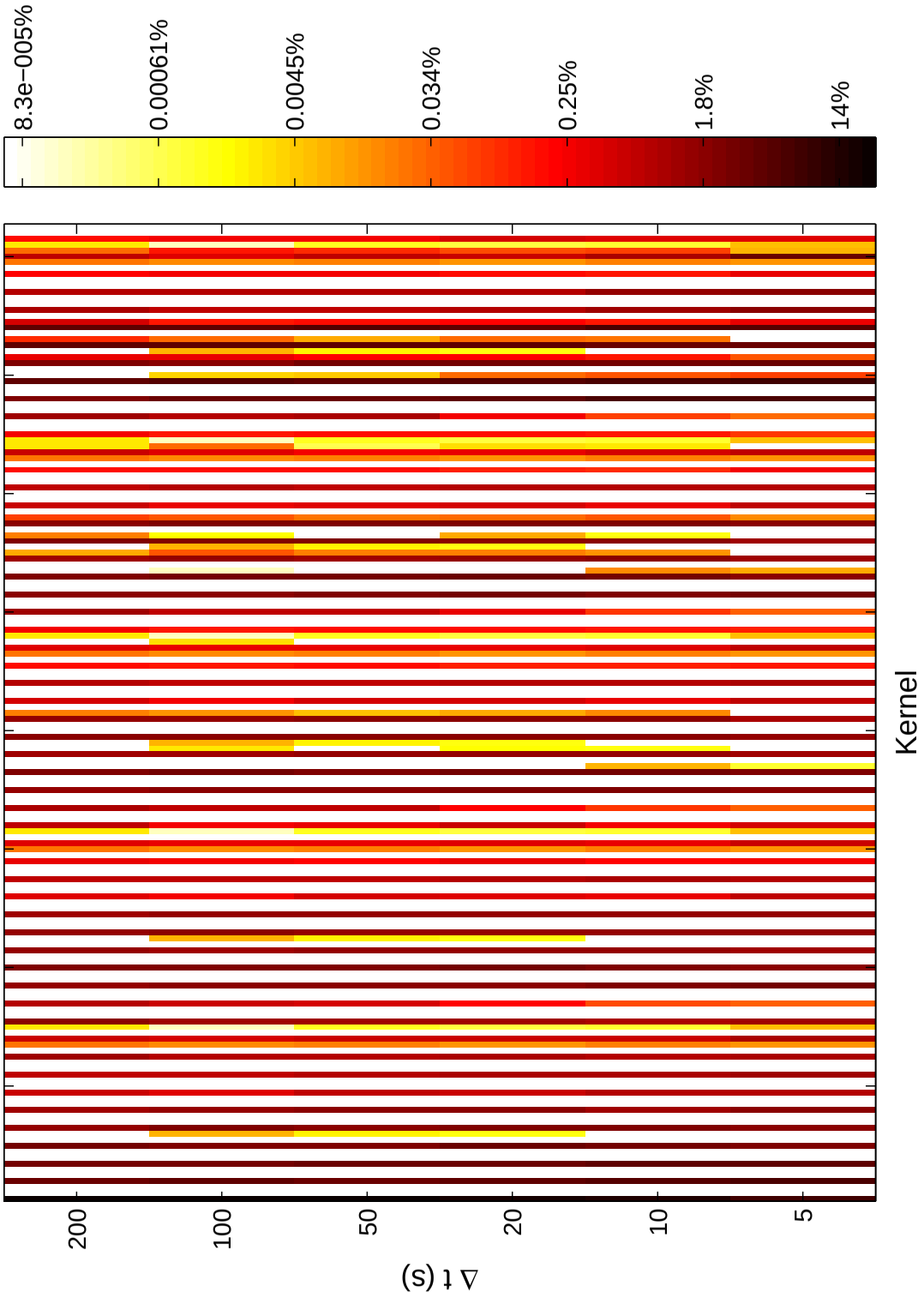


Figure 42: Relationship between % reduction in β and % accuracy for full MKL set

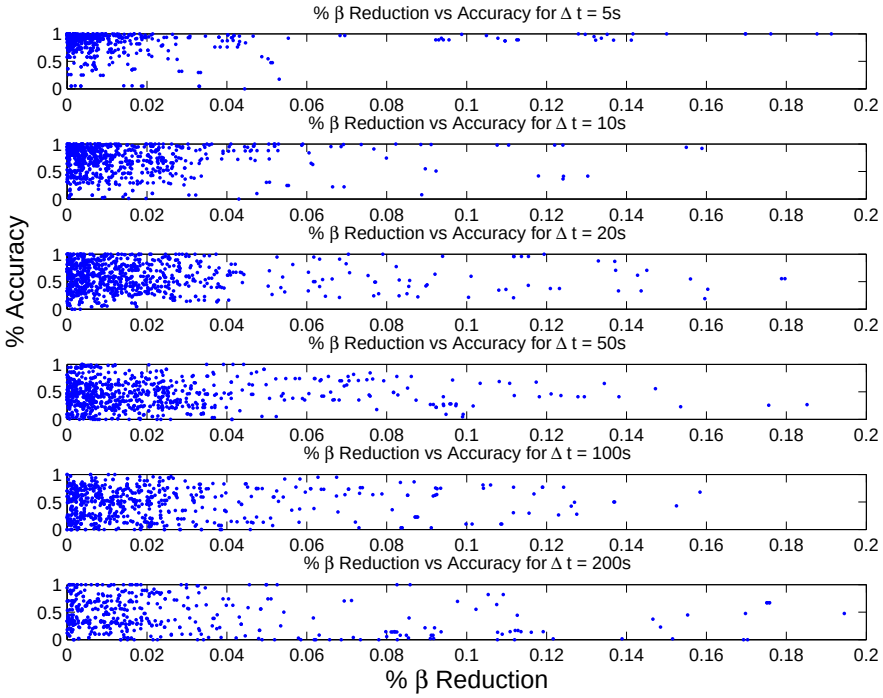


Figure 43: Relationship between % reduction in β and % accuracy for Subset A

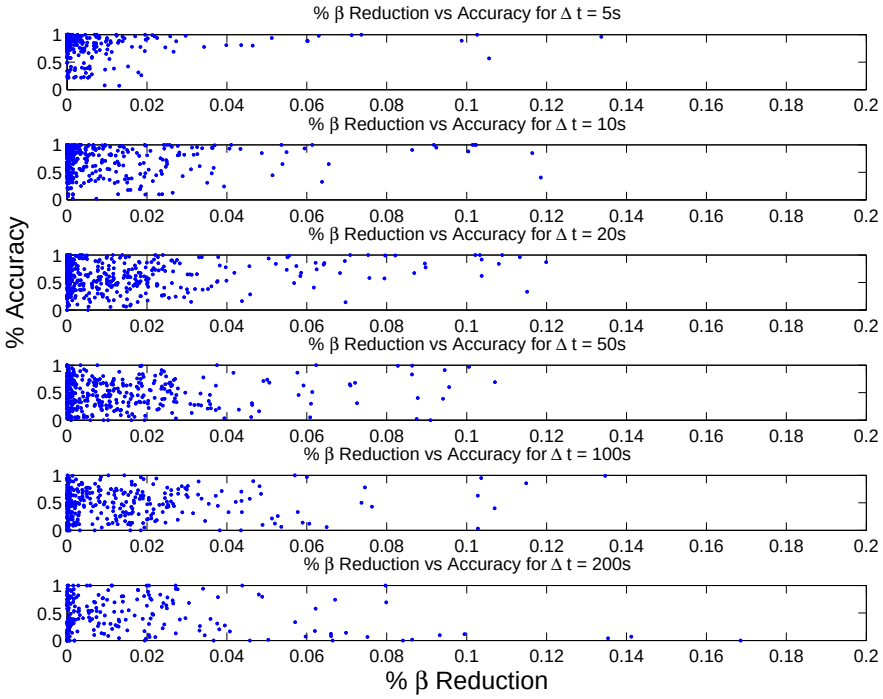


Figure 44: Relationship between % reduction in β and % accuracy for Subset B

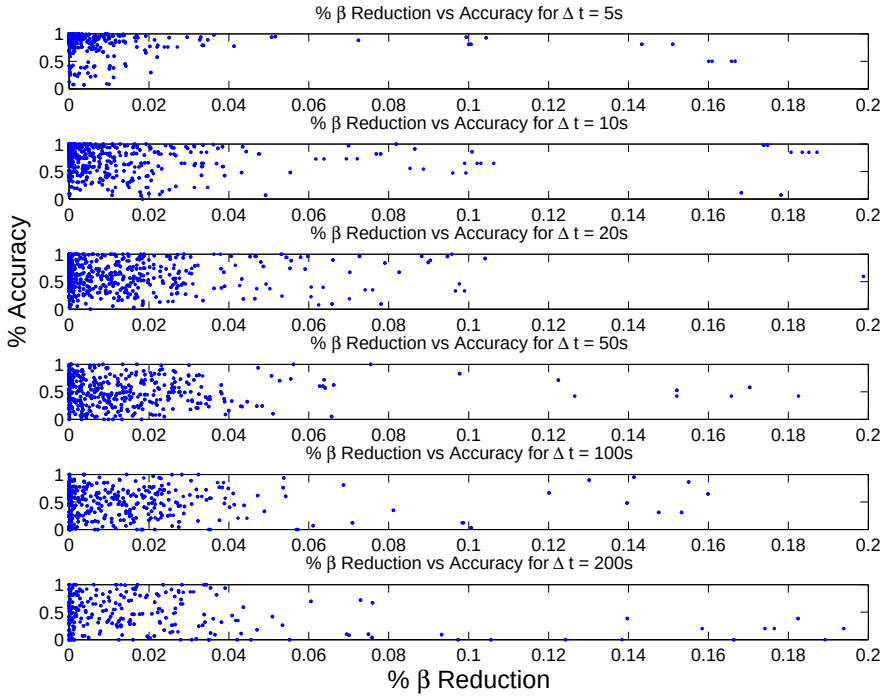


Figure 45: Relationship between final β value and % accuracy for full MKL set

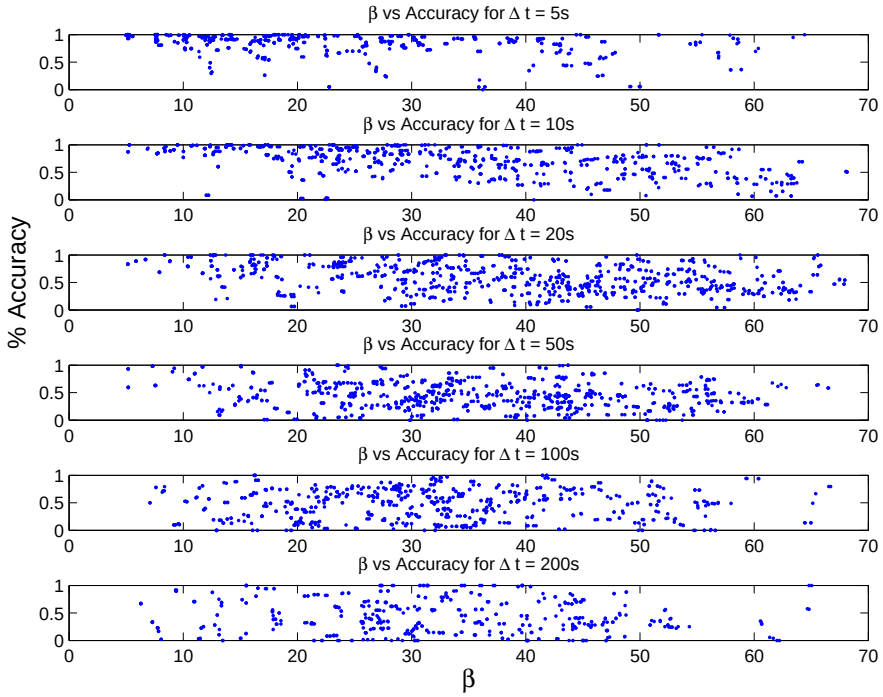


Figure 46: Relationship between final β value and % accuracy for Subset A

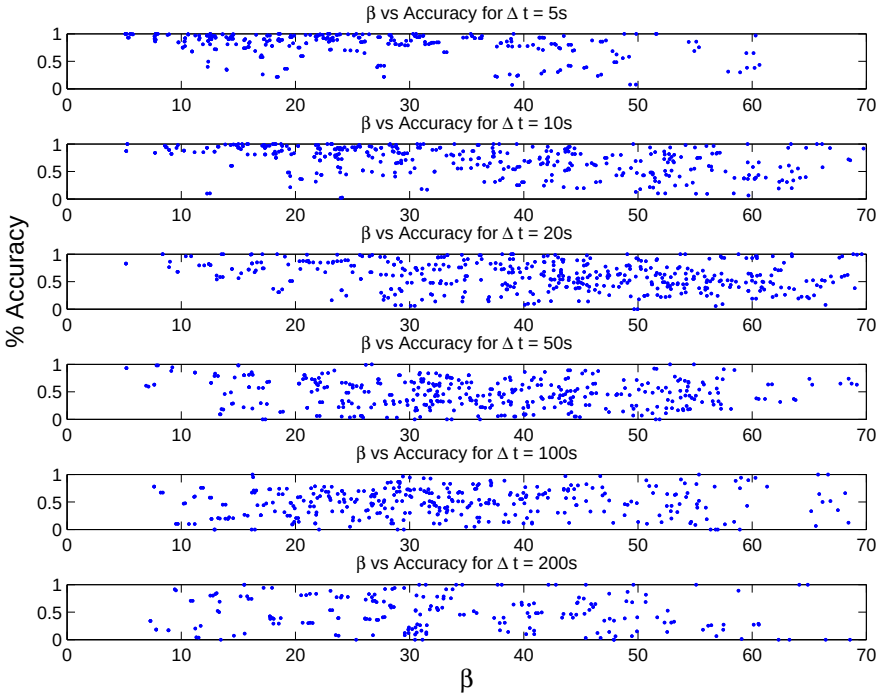


Figure 47: Relationship between final β value and % accuracy for Subset B

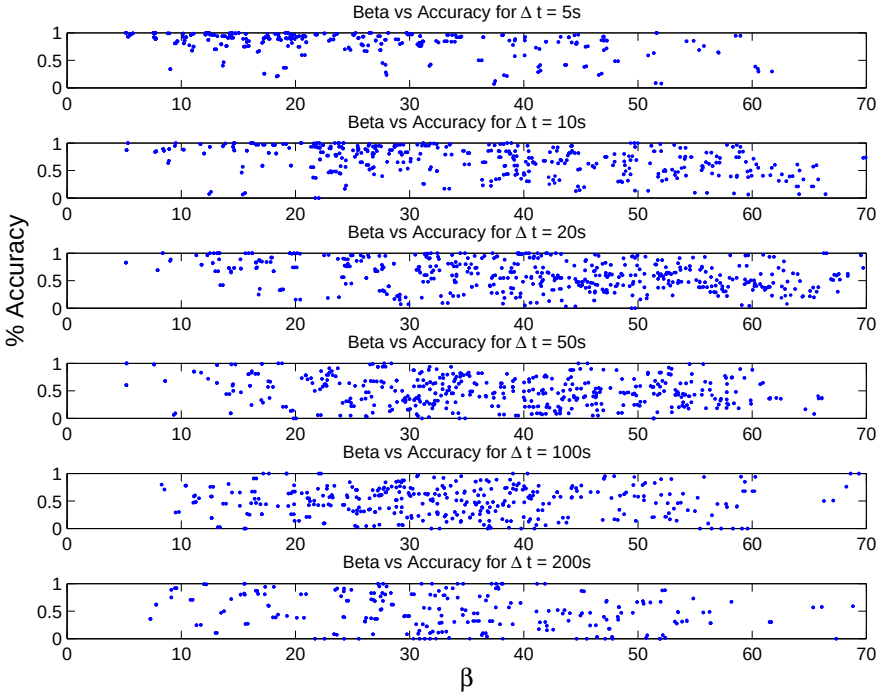


Figure 48: RBF hyperparam. values for full MKL set with Serial Hyperparam. and Fisher Optimisation

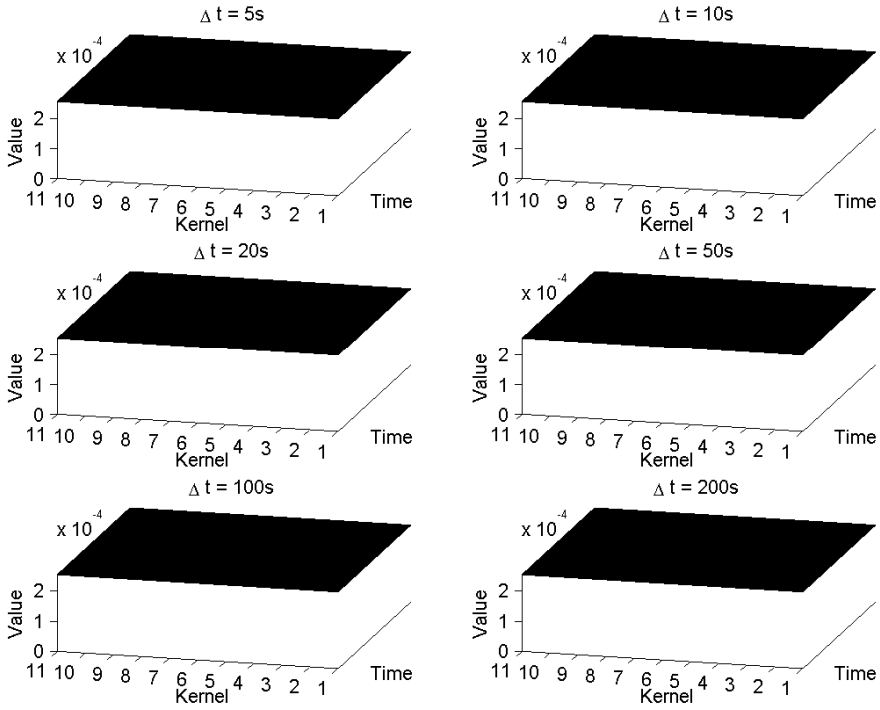


Figure 49: RBF hyperparam. values for full MKL set with Parallel Hyperparam. and Fisher Optimisation

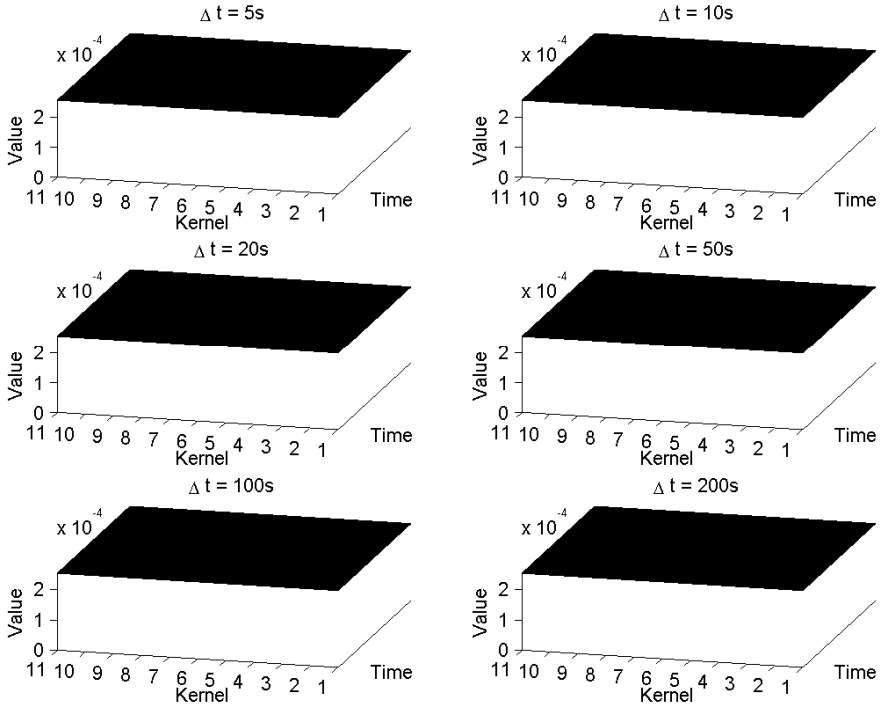


Figure 50: RBF hyperparam. values for full MKL set with Serial Fisher Optimisation

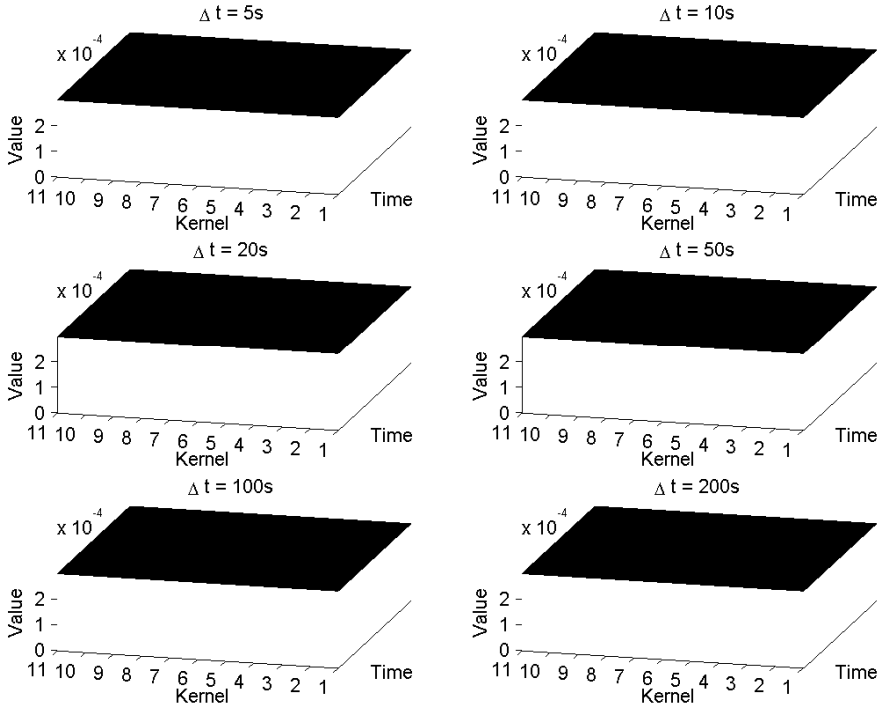


Figure 51: RBF hyperparam. values for full MKL set with Parallel Fisher Optimisation

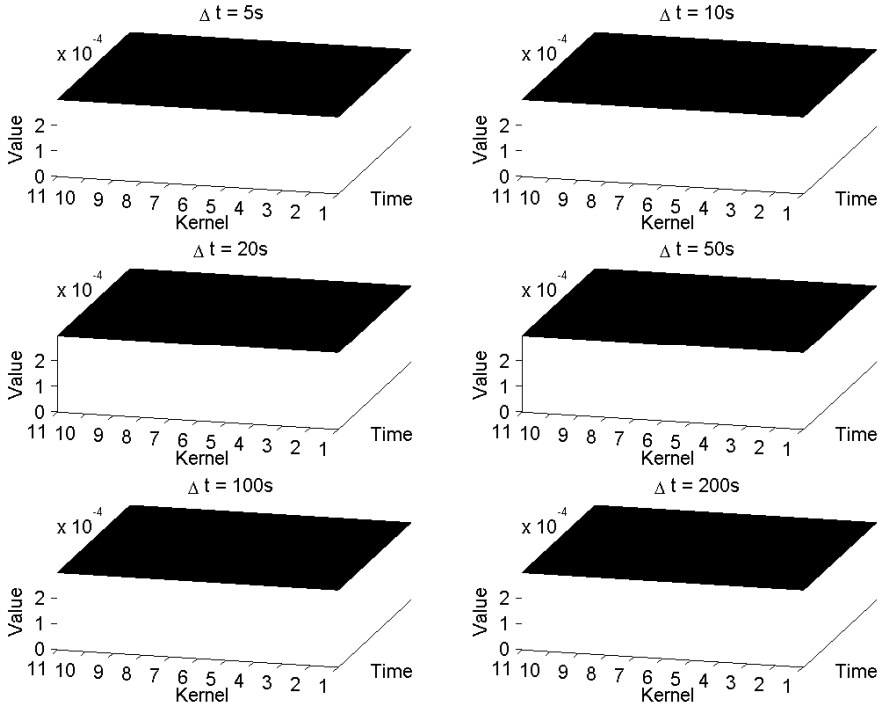


Figure 52: Poly hyperparam. values for full MKL set with Serial Hyperparam. and Fisher Optimisation

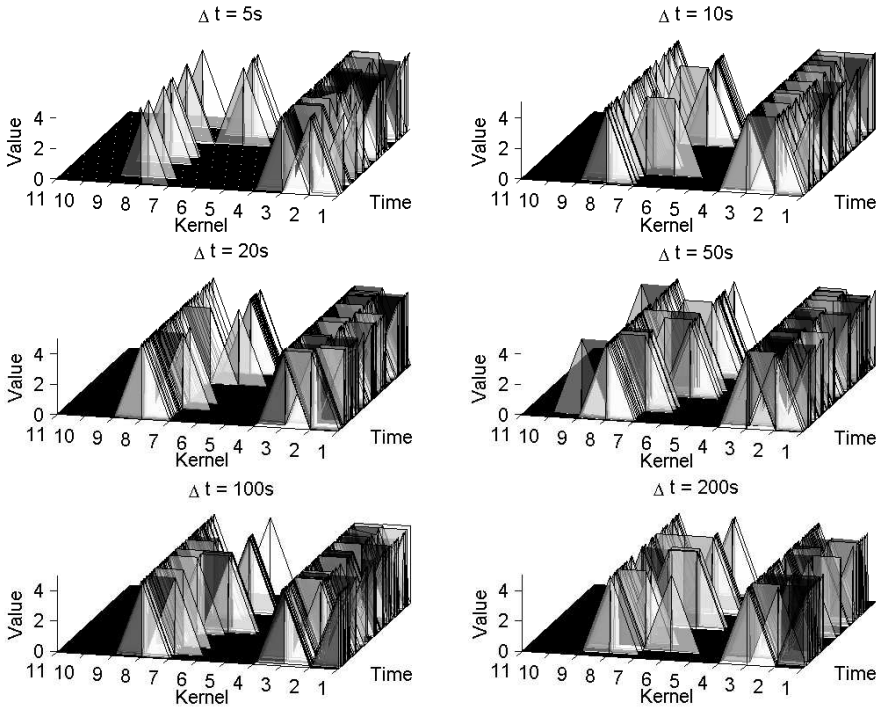


Figure 53: Poly hyperparam. values for full MKL set with Parallel Hyperparam. and Fisher Optimisation

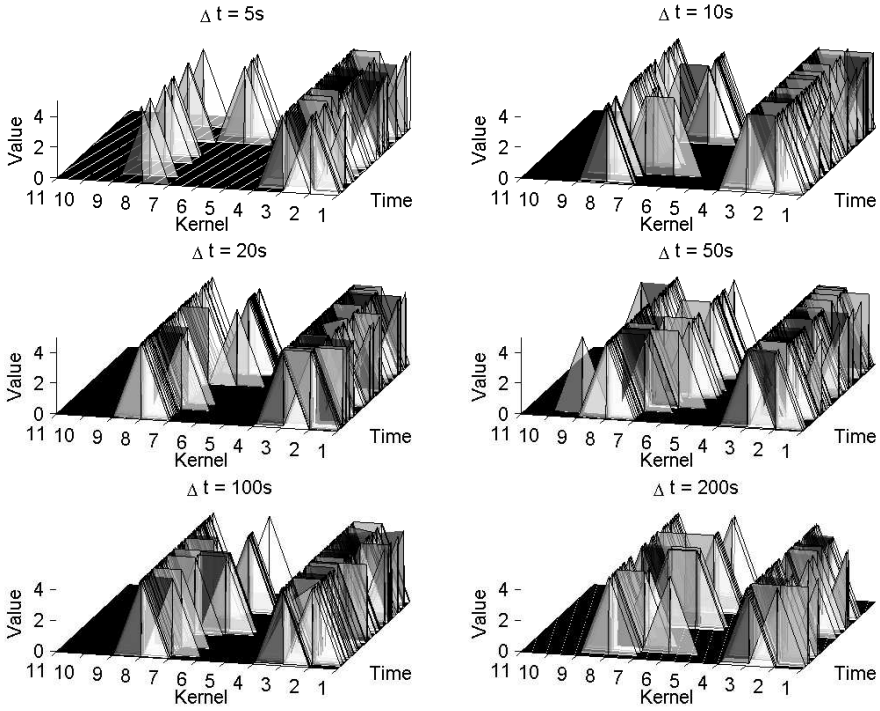


Figure 54: Poly hyperparam. values for full MKL set with Serial Fisher Optimisation

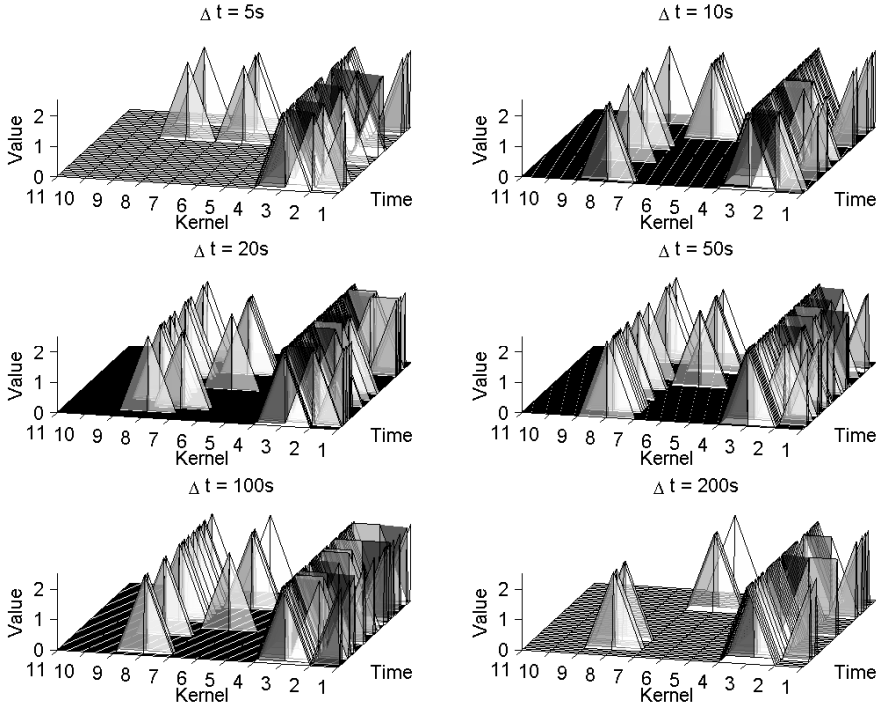


Figure 55: Poly hyperparam. values for full MKL set with Parallel Fisher Optimisation

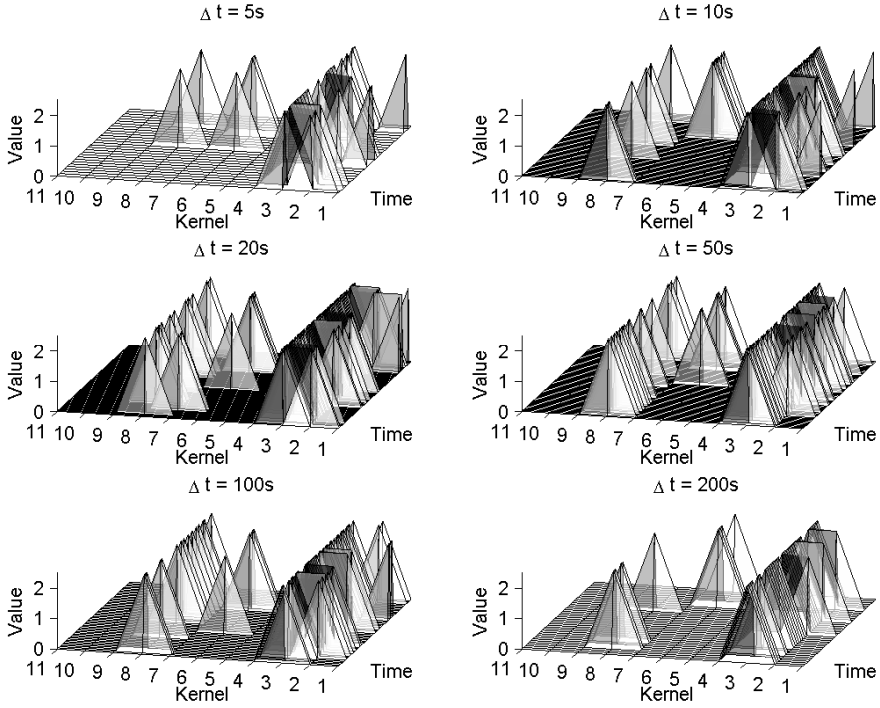


Figure 56: INN hyperparam. values for full MKL set with Serial Hyperparam. and Fisher Optimisation

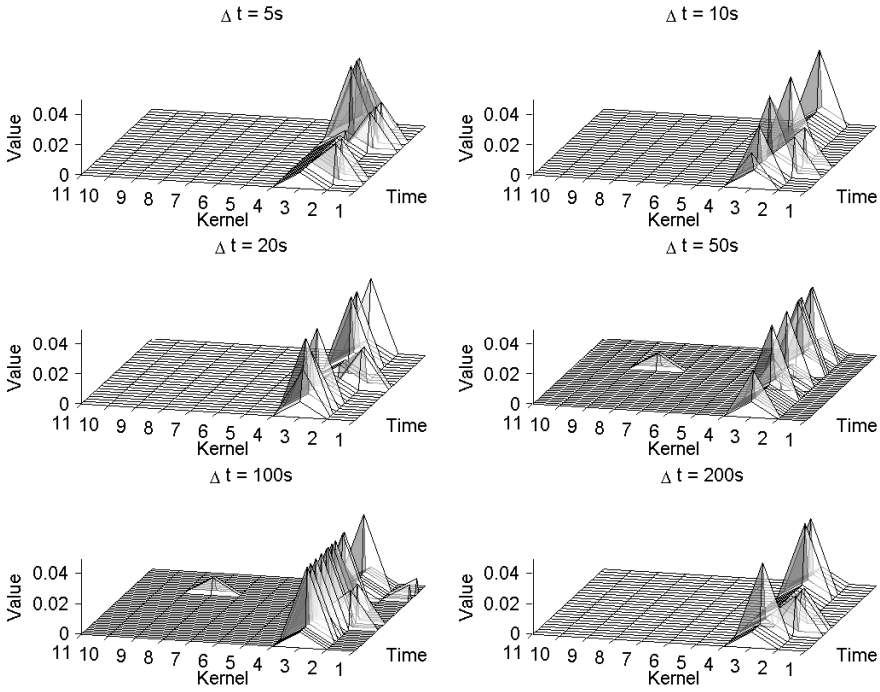


Figure 57: INN hyperparam. values for full MKL set with Parallel Hyperparam. and Fisher Optimisation

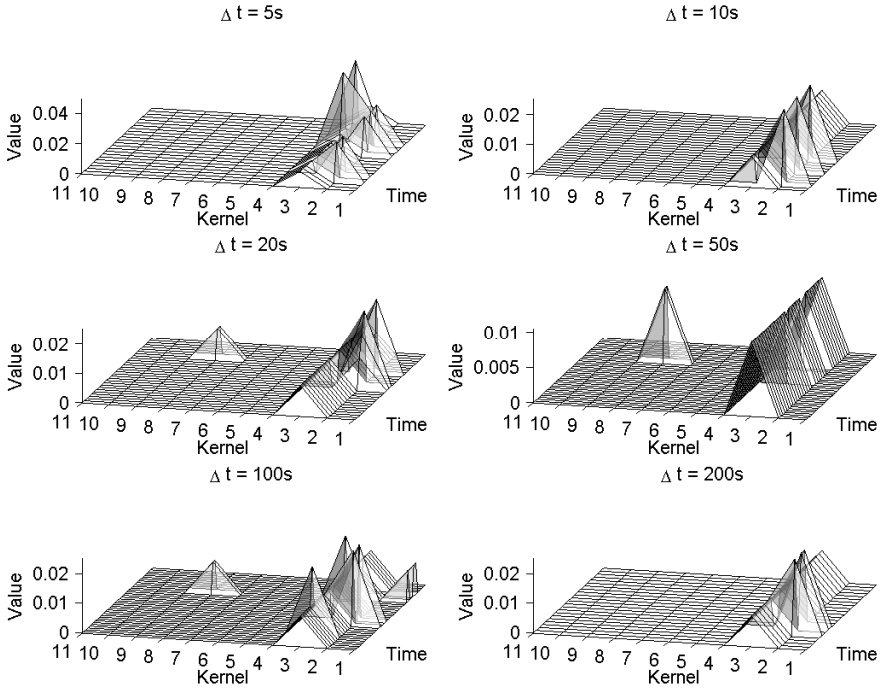


Figure 58: INN hyperparam. values for full MKL set with Serial Fisher Optimisation

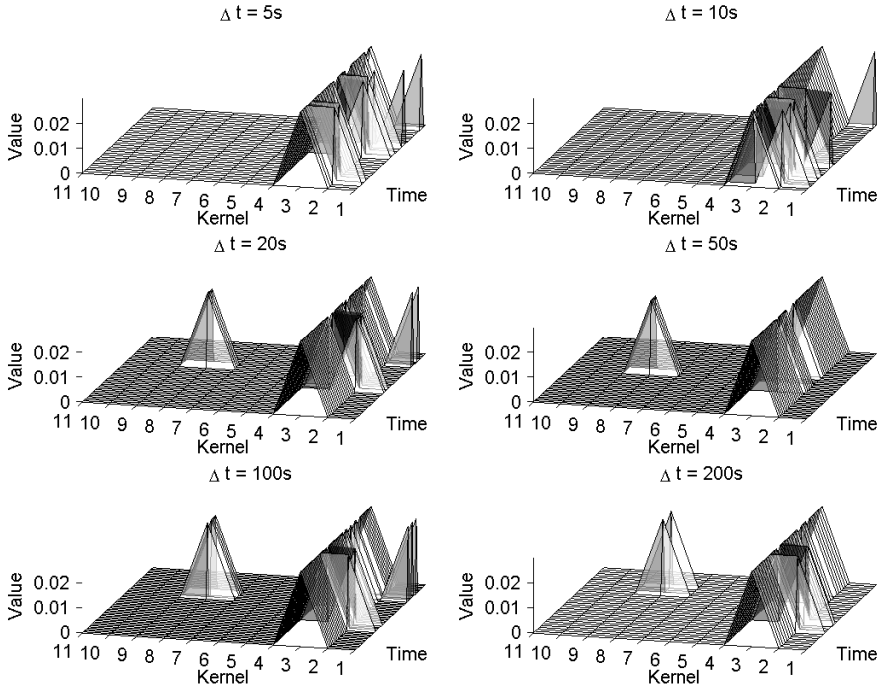


Figure 59: INN hyperparam. values for full MKL set with Parallel Fisher Optimisation

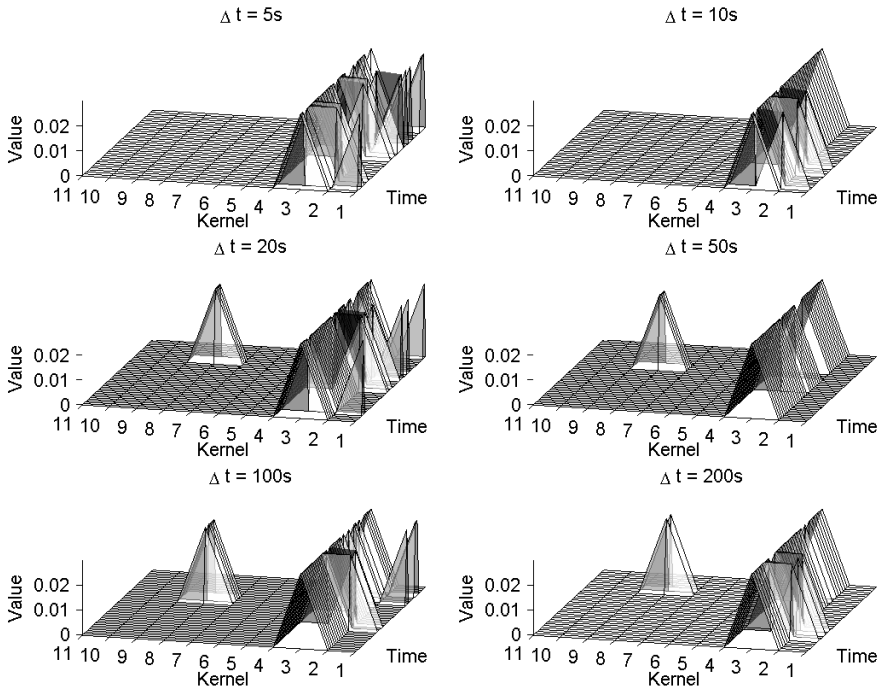


Figure 60: Time series of parameters w_{Bid} and w_{Ask} from ACD model

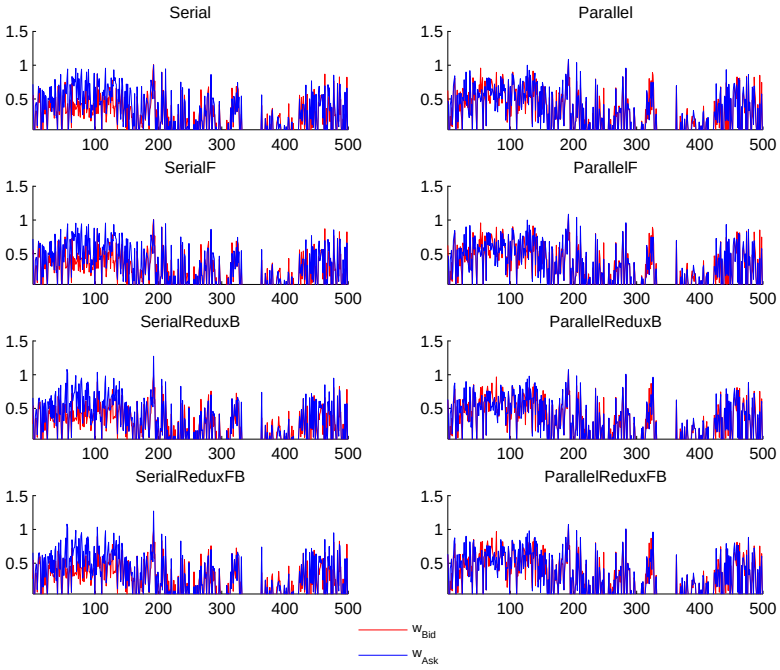


Figure 61: Time series of parameters q_{Bid} and q_{Ask} from ACD model

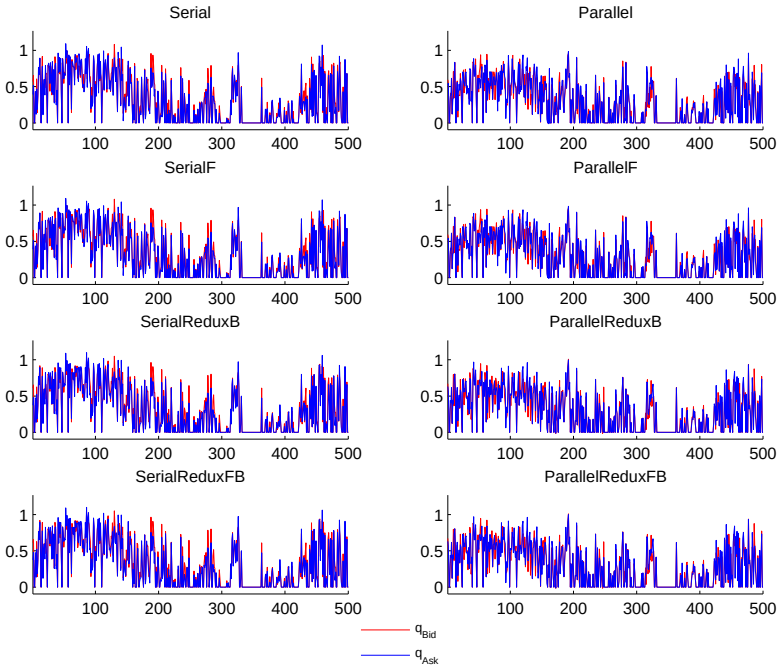


Figure 62: Time series of parameters p_{Bid} and p_{Ask} from ACD model

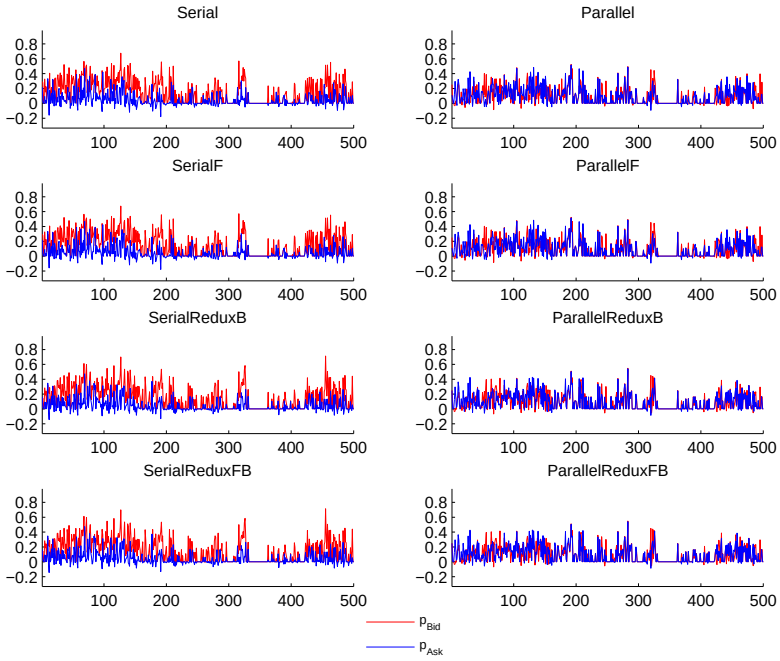


Figure 63: Time series of parameters L_{Bid} and L_{Ask} from ACD model

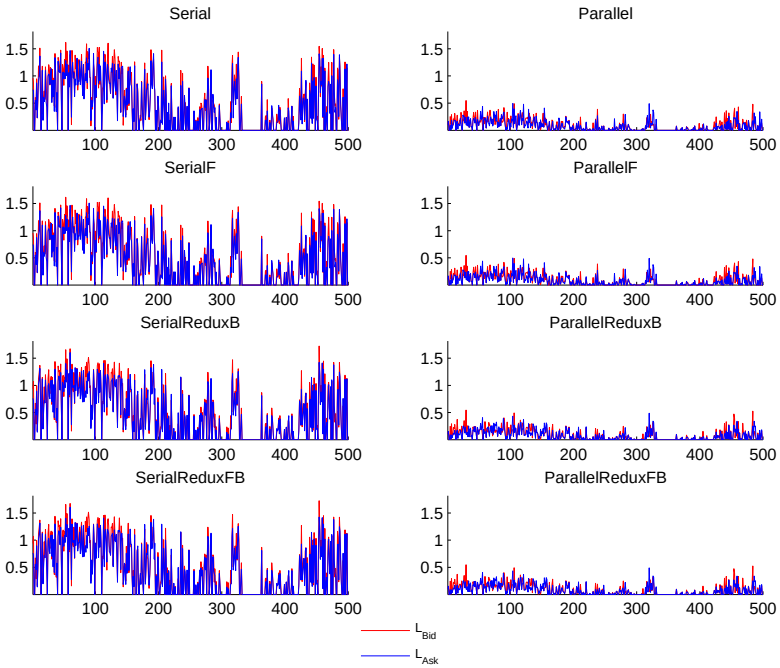


Figure 64: Time series of parameters λ_1^{Bid} and λ_1^{Ask} from Poisson model

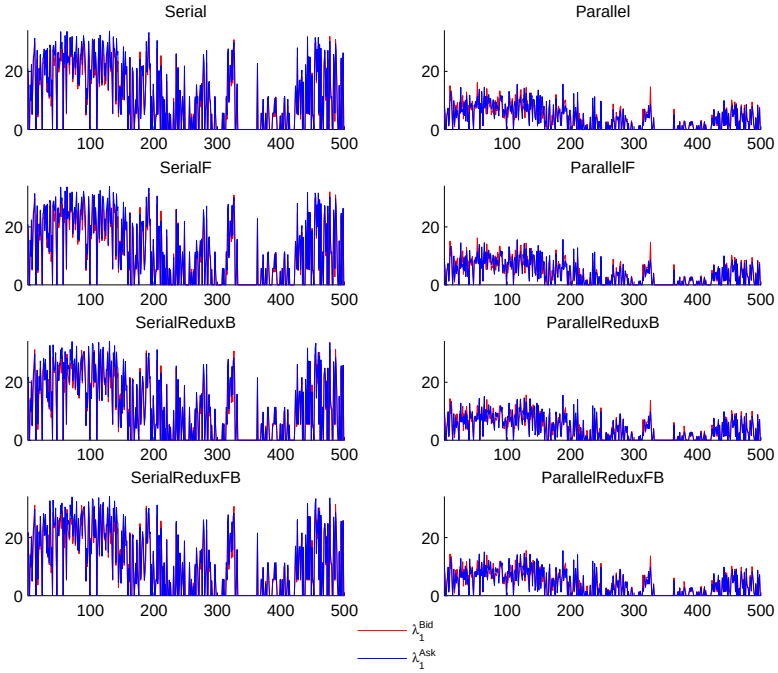


Figure 65: Time series of parameters λ_2^{Bid} and λ_2^{Ask} from Poisson model

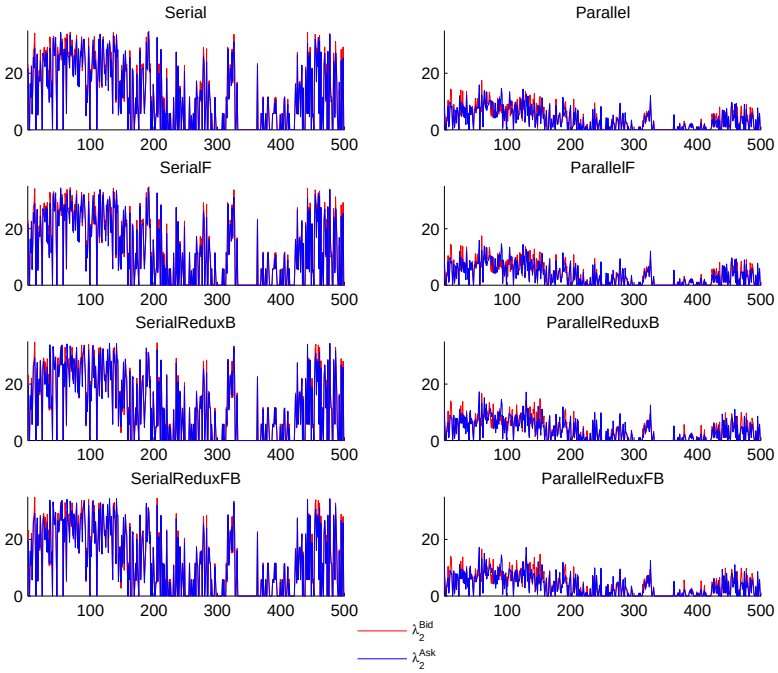


Figure 66: Time series of parameters λ_3^{Bid} and λ_3^{Ask} from Poisson model

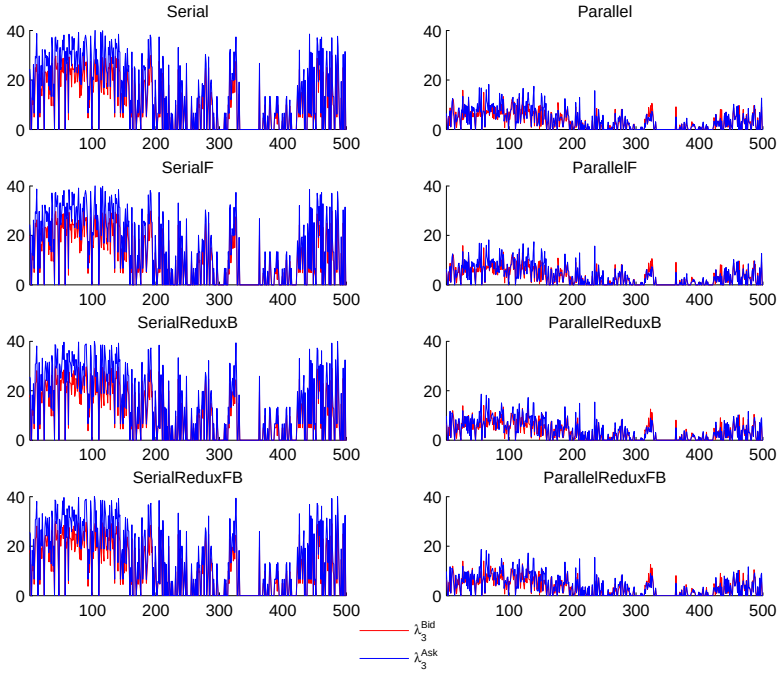


Figure 67: Time series of parameters λ_4^{Bid} and λ_4^{Ask} from Poisson model

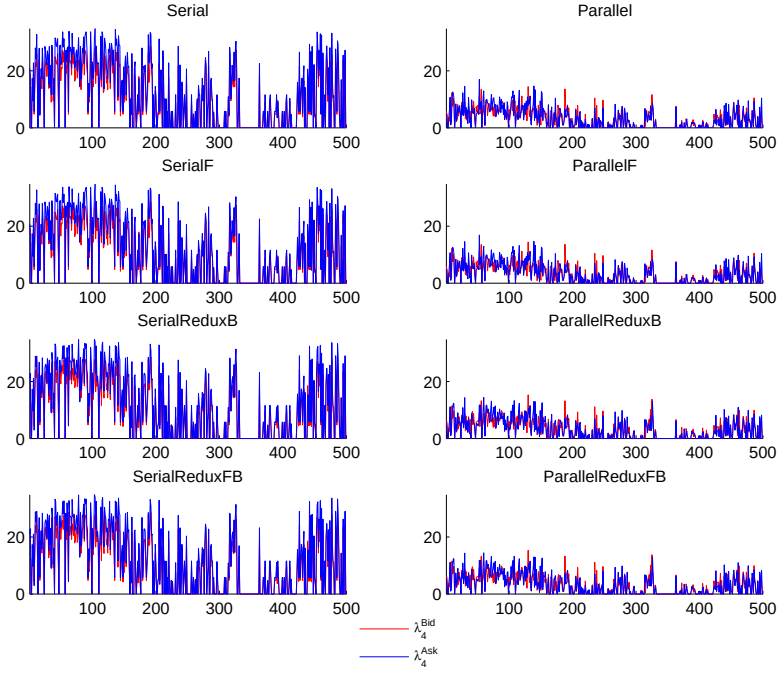


Figure 68: Time series of parameters λ_5^{Bid} and λ_5^{Ask} from Poisson model

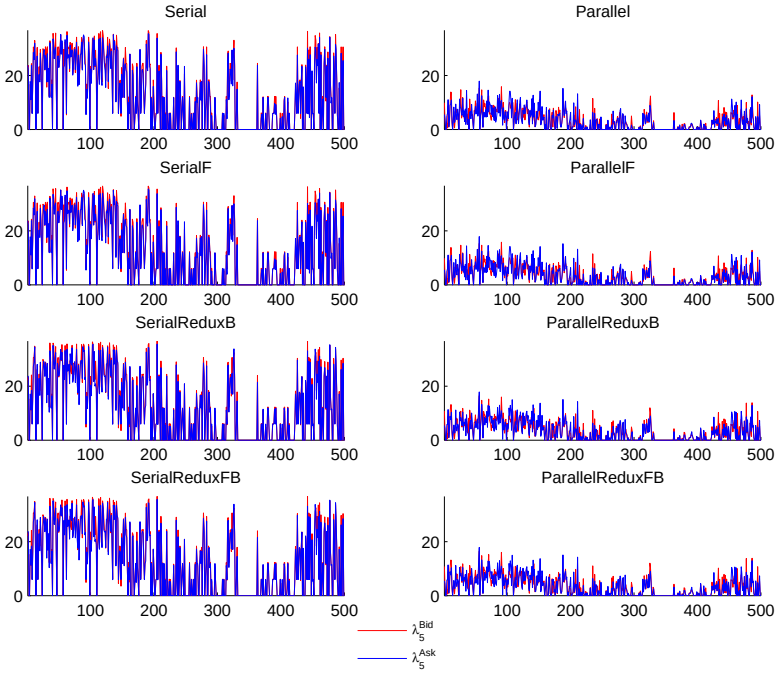


Figure 69: Time series of parameters μ_{Bid} and μ_{Ask} from Wiener model

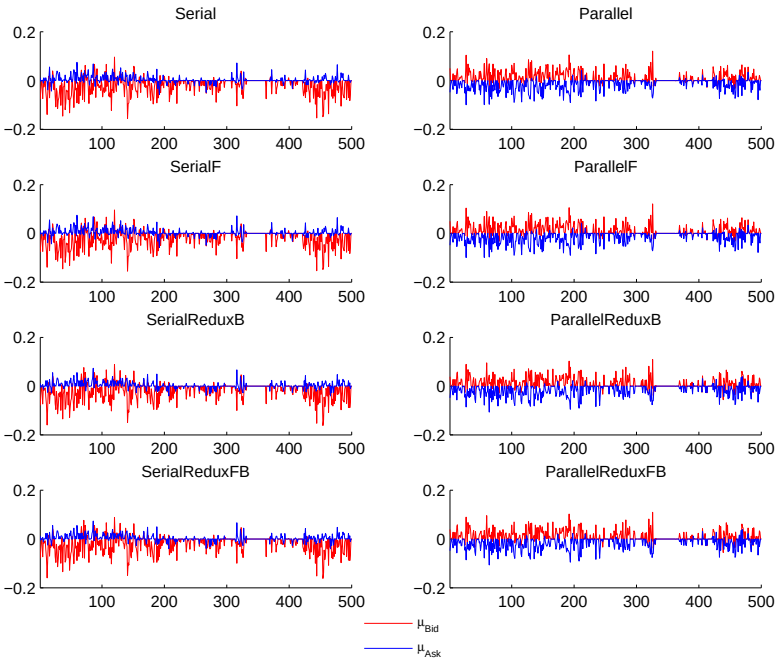


Figure 70: Time series of parameters σ_{Bid} and σ_{Ask} from Wiener model

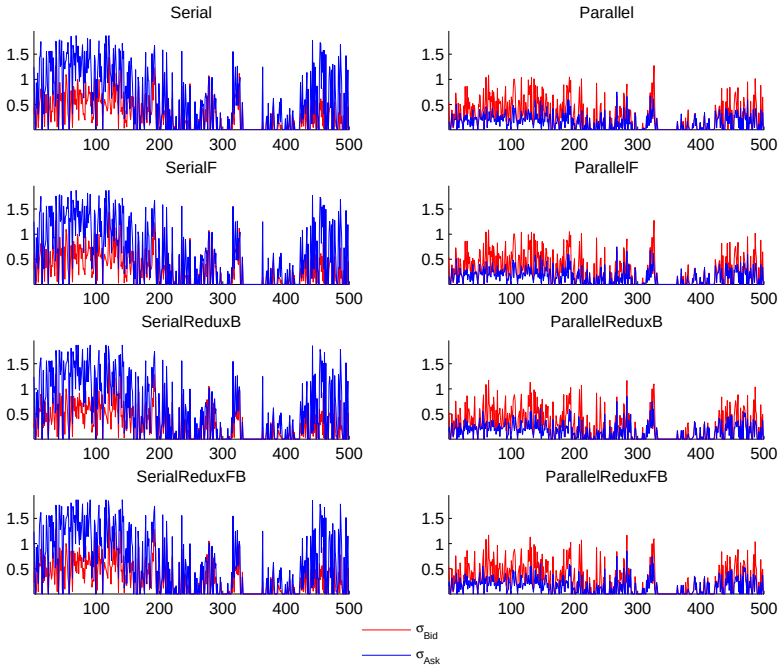


Figure 71: Comparison between learnt and ML estimates for w_{Bid} and w_{Ask} from ACD model

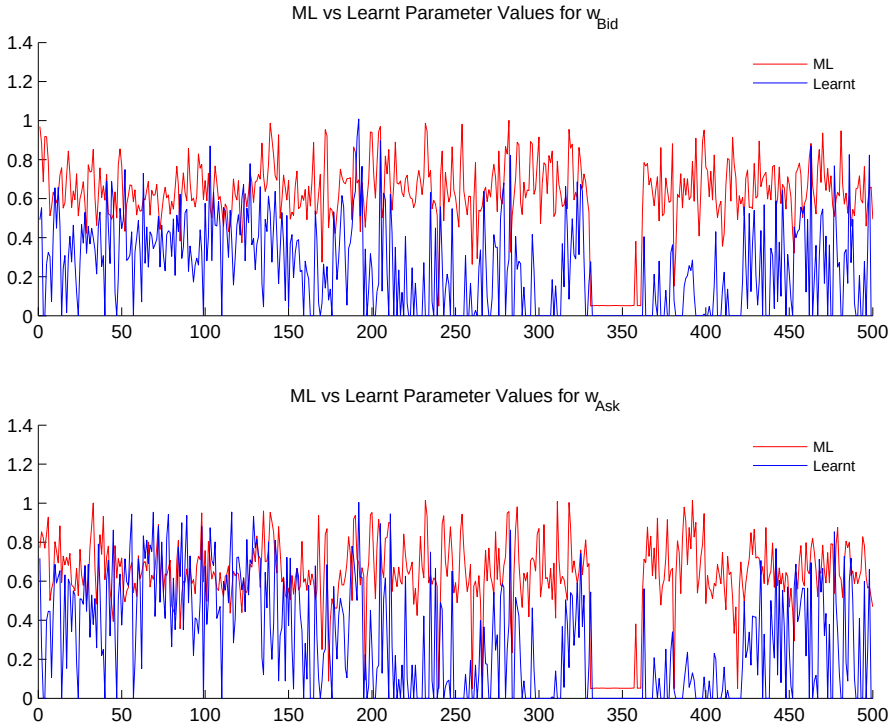


Figure 72: Comparison between learnt and ML estimates for q_{Bid} and q_{Ask} from ACD model

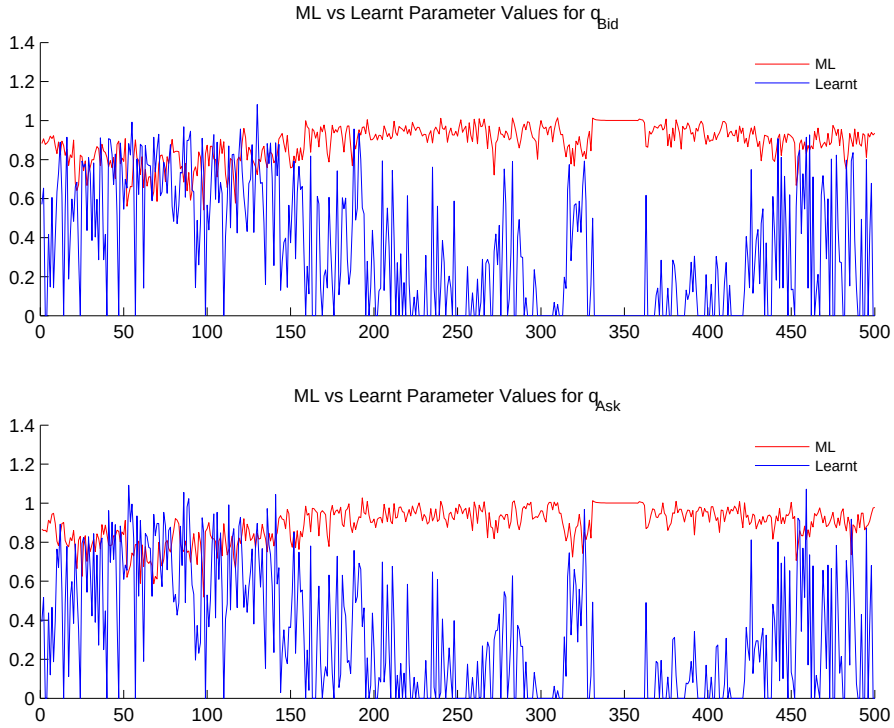


Figure 73: Comparison between learnt and ML estimates for p_{Bid} and p_{Ask} from ACD model

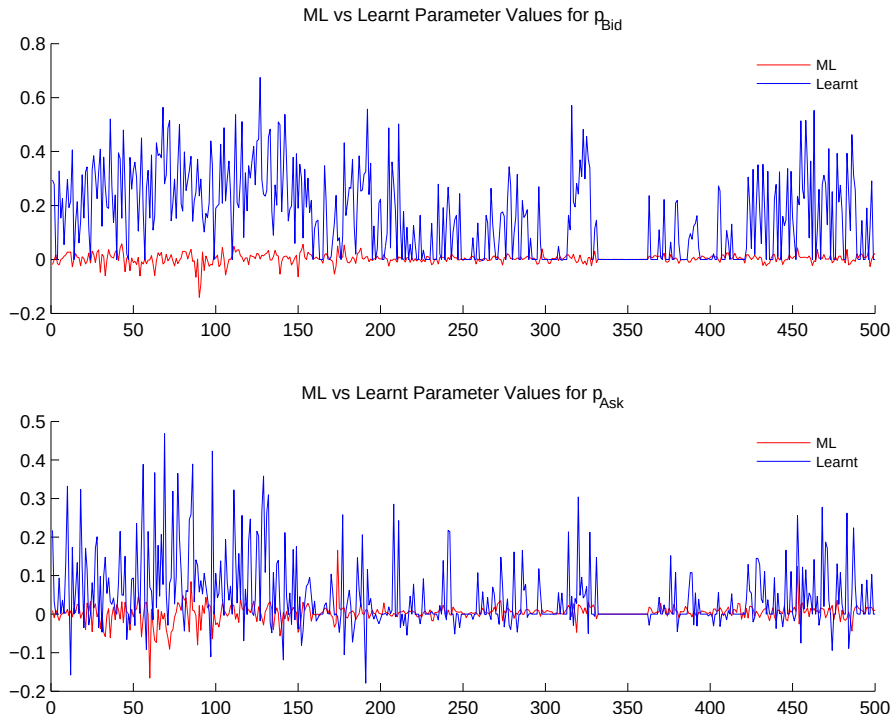


Figure 74: Comparison between learnt and ML estimates for L_{Bid} and L_{Ask} from ACD model

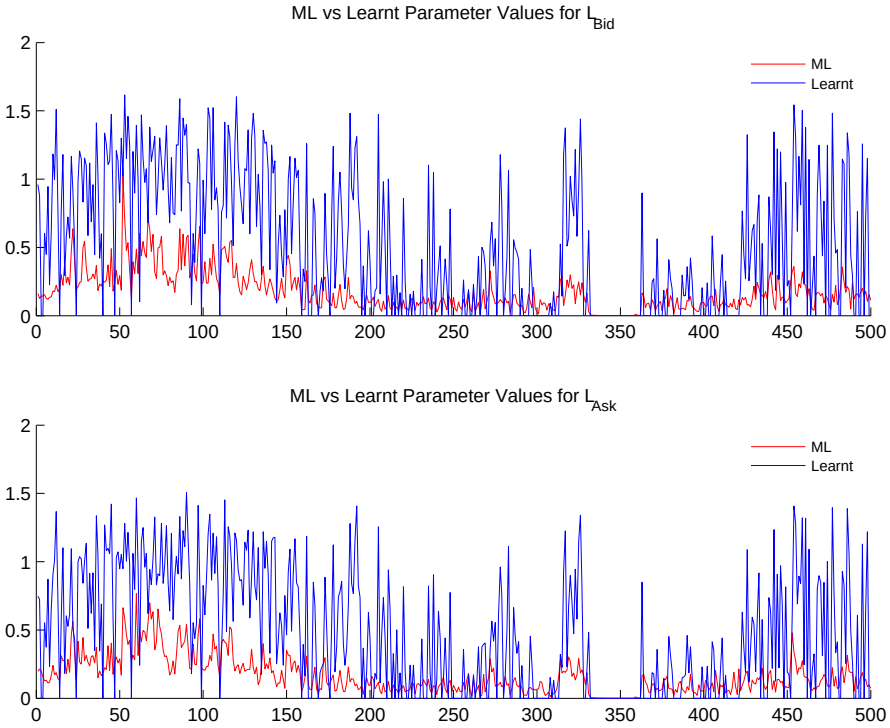


Figure 75: Comparison between learnt and ML estimates for λ_1^{Bid} and λ_1^{Ask} from Poisson model

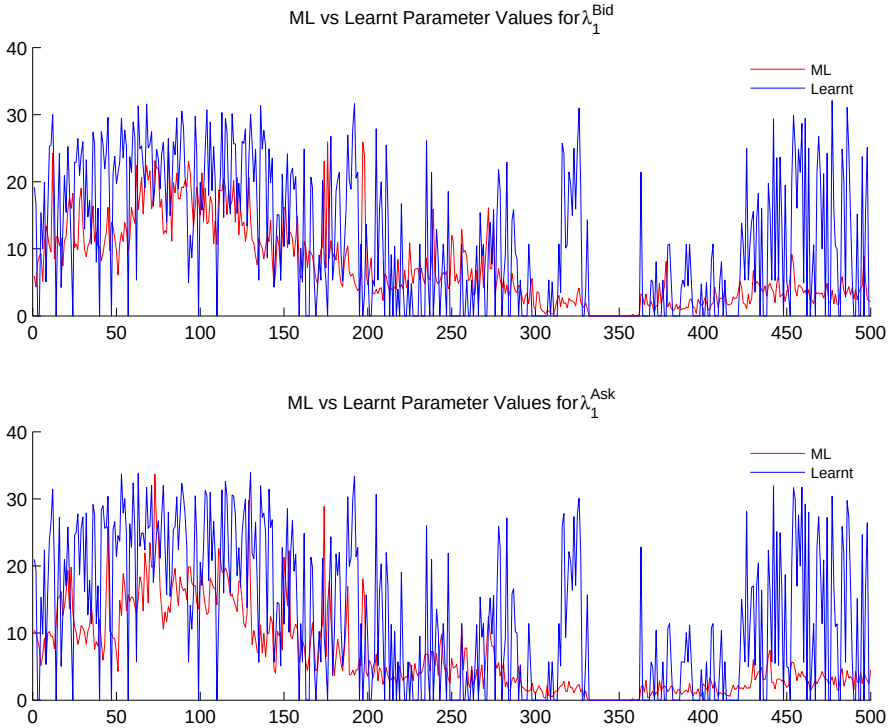


Figure 76: Comparison between learnt and ML estimates for λ_2^{Bid} and λ_2^{Ask} from Poisson model

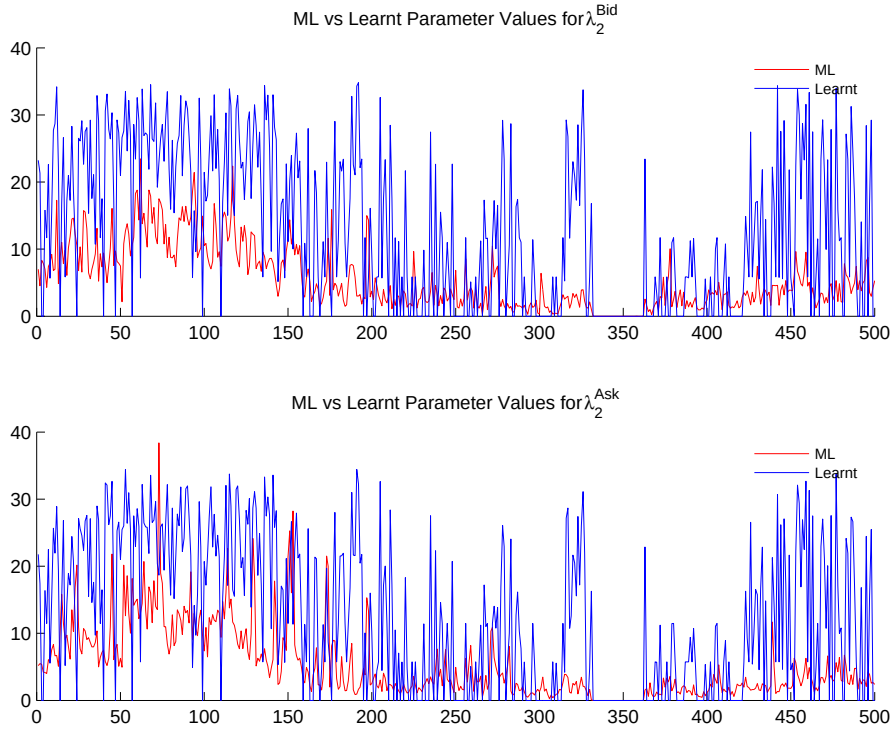


Figure 77: Comparison between learnt and ML estimates for λ_3^{Bid} and λ_3^{Ask} from Poisson model

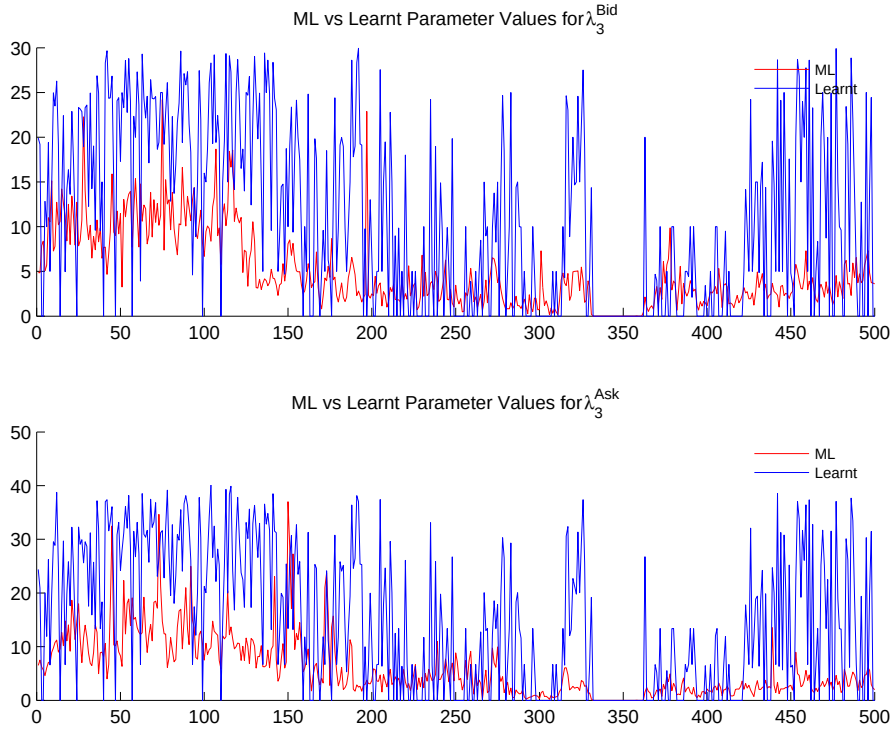


Figure 78: Comparison between learnt and ML estimates for λ_4^{Bid} and λ_4^{Ask} from Poisson model

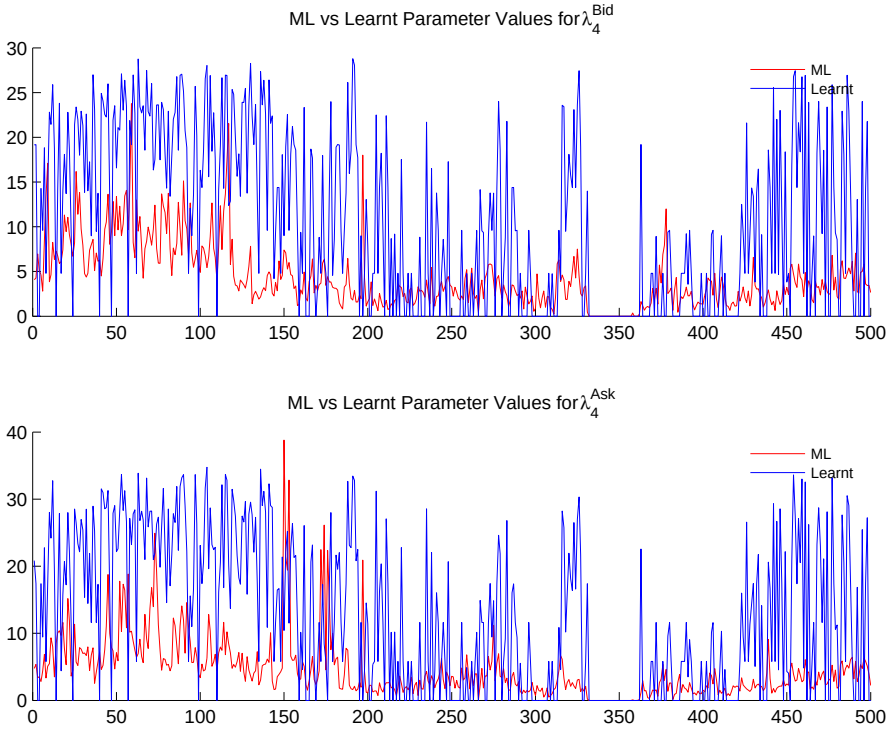


Figure 79: Comparison between learnt and ML estimates for λ_5^{Bid} and λ_5^{Ask} from Poisson model

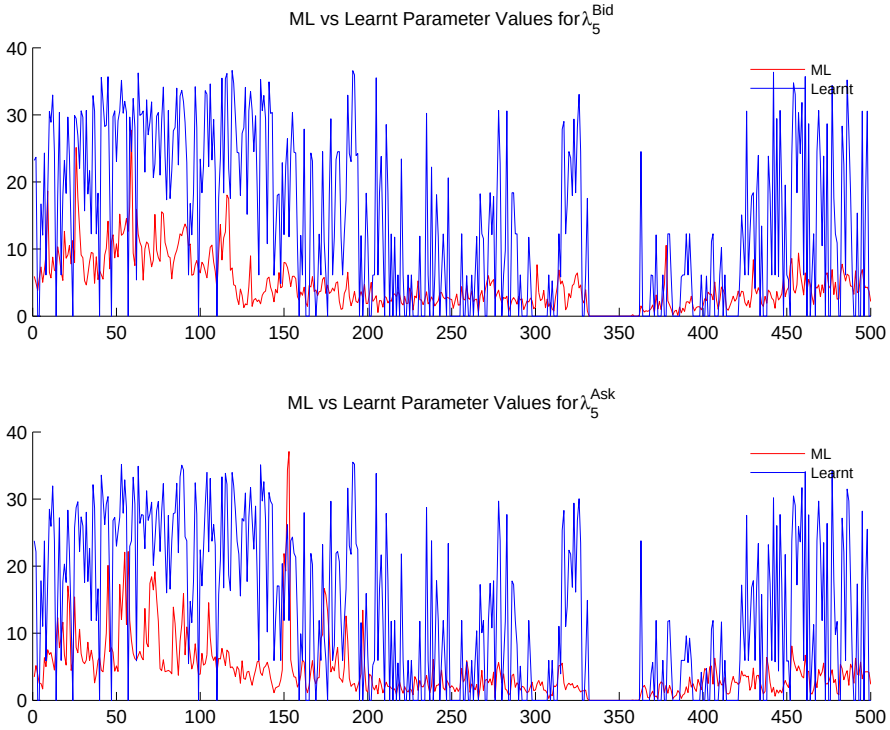


Figure 80: Comparison between learnt and ML estimates for μ_{Bid} and μ_{Ask} from Wiener model

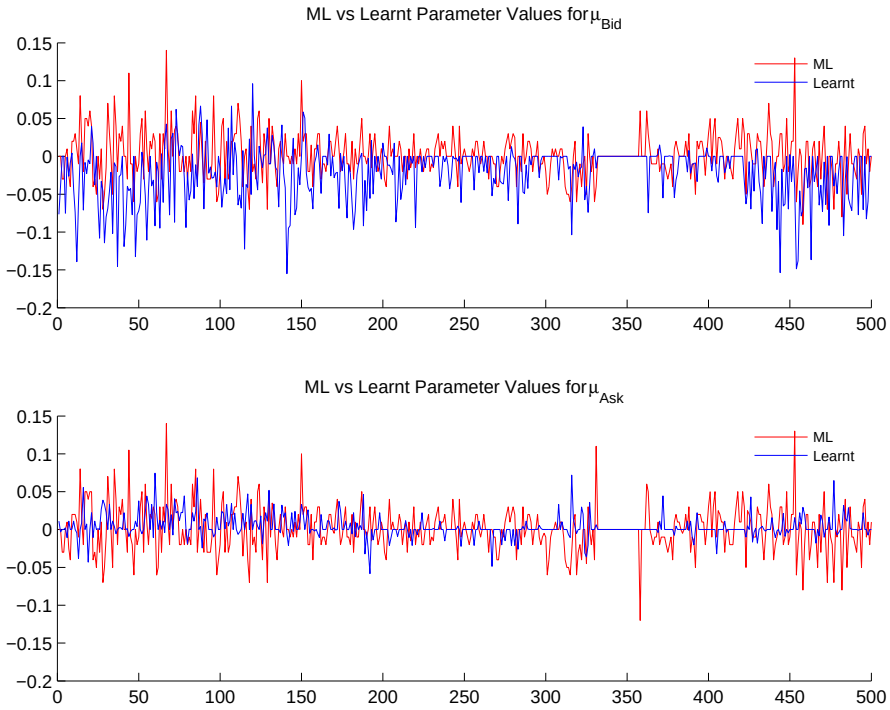
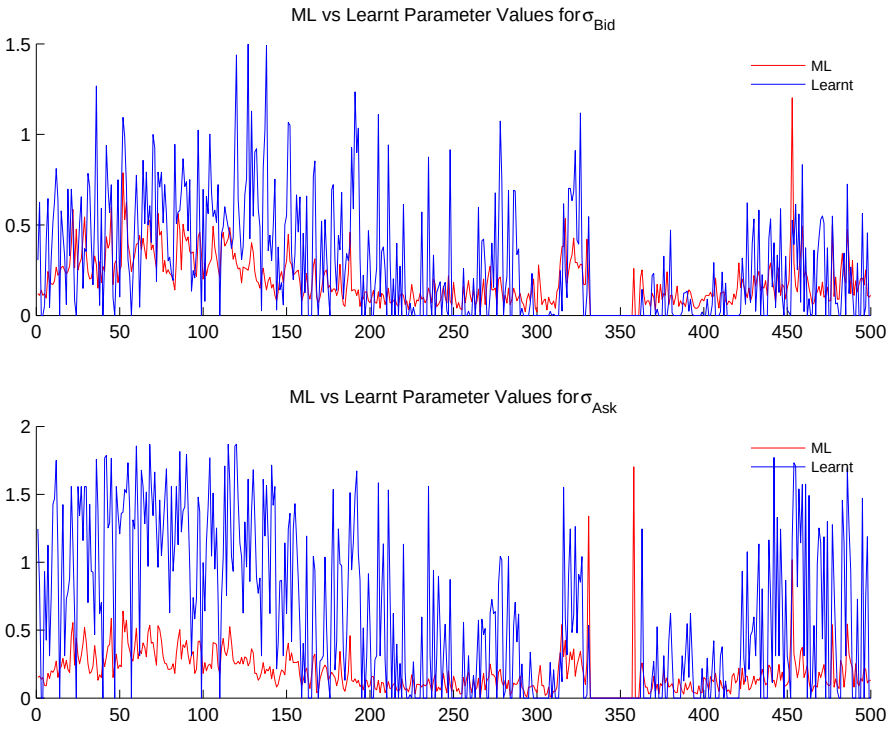


Figure 81: Comparison between learnt and ML estimates for σ_{Bid} and σ_{Ask} from Wiener model



17.8 Tables of results

Table 15: % accuracy of individual SVM by feature type

Feature	Δt					
	5	**10**	**20**	**50**	**100**	**200**
F_1	60.5	48.4	38.2	38.4	33.3	42.7
F_2	62.6	58.4	50.5	39.5	32.0	43.8
F_3	58.1	52.2	46.4	34.7	36.0	29.5
F_4	59.2	56.0	52.5	38.7	33.8	41.2
F_5	60.0	59.4	48.1	37.5	35.3	39.5
F_6	61.6	57.4	48.0	39.3	39.8	38.2
F_7	64.8	57.9	50.3	43.9	39.8	35.9
F_8	70.7	59.1	51.2	43.5	38.5	37.4
F_9	64.0	58.2	46.5	38.8	37.7	36.5
F_{10}	67.8	59.7	47.3	36.7	34.9	33.3
F_{11}	55.5	46.2	47.0	40.2	34.5	38.8

Table 16: % accuracy of individual SVM by kernel mapping

Mapping	Δt					
	5	**10**	**20**	**50**	**100**	**200**
RBF	83.0	74.7	64.0	45.5	39.6	42.4
Poly	50.3	45.9	40.6	36.9	34.5	35.5
INN	53.5	46.5	38.9	35.2	33.8	35.7

Table 17: % accuracy of individual SVM by feature class

Class	Δt					
	5	**10**	**20**	**50**	**100**	**200**
Price	60.1	53.8	46.9	37.8	33.8	39.3
Volume	64.3	58.4	49.4	41.1	38.4	37.8
Fisher	62.4	54.7	46.9	38.6	35.7	36.2

Table 18: Stability of individual SVM by feature type

Feature	Δt					
	5	**10**	**20**	**50**	**100**	**200**
F_1	6.4	8.6	12.6	15.4	19.9	20.6
F_2	7.4	8.1	10.6	15.0	16.0	19.0
F_3	8.1	9.1	11.3	19.9	20.7	18.6
F_4	8.5	9.8	12.7	15.1	17.5	19.8
F_5	7.7	9.3	12.6	17.8	20.7	18.7
F_6	7.6	8.7	11.1	21.7	20.4	21.4
F_7	7.6	7.5	10.9	16.9	19.0	24.2
F_8	8.1	7.3	10.3	15.6	21.3	23.7
F_9	6.8	7.5	10.7	17.0	20.0	24.9
F_{10}	10.2	8.4	11.0	15.3	18.7	30.0
F_{11}	6.2	9.8	12.9	20.2	22.7	46.4
BM	159.7	51.7	19.4	10.7	8.8	6.9

Table 19: Stability of individual SVM by kernel mapping

Mapping	Δt					
	5	**10**	**20**	**50**	**100**	**200**
RBF	26.7	7.3	8.7	14.1	17.9	21.9
Poly	5.4	9.1	13.6	19.6	19.9	23.7
INN	5.7	9.3	13.7	18.3	21.1	22.6

Table 20: Stability of individual SVM by feature class

Class	Δt					
	5	**10**	**20**	**50**	**100**	**200**
Price	7.5	8.8	11.7	16.1	18.3	19.5
Volume	7.8	8.1	11.2	17.7	20.3	21.8
Fisher	7.4	8.5	11.5	17.3	20.3	31.6

Table 21: Correlation of MKL weighting & number of times included vs % accuracy

	Δt					
	5	**10**	**20**	**50**	**100**	**200**
Mean Weight	0.59	0.53	0.46	0.38	0.33	0.33
Num Times	0.88	0.91	0.92	0.73	0.51	0.50

Table 22: Average weighting of individual SVM by feature type

Feature	Δt					
	5	**10**	**20**	**50**	**100**	**200**
F_1	7.7	13.0	18.3	24.5	25.4	27.4
F_2	20.2	17.9	14.8	12.7	13.3	10.6
F_3	18.9	18.6	16.9	15.0	16.7	14.4
F_4	9.8	11.0	11.5	9.7	8.7	9.8
F_5	9.5	11.3	11.8	12.3	12.9	11.5
F_6	12.0	13.4	12.8	13.0	11.4	11.2
F_7	5.0	3.0	2.7	2.7	1.9	2.9
F_8	6.1	5.5	4.3	3.9	3.7	3.9
F_9	2.1	1.5	1.8	1.8	1.8	2.2
F_{10}	6.4	2.9	2.3	2.3	2.2	2.7
F_{11}	2.2	2.0	2.7	2.1	2.0	3.4

Table 23: Average weighting of individual SVM by kernel mapping

Mapping	Δt					
	5	**10**	**20**	**50**	**100**	**200**
RBF	99.5	99.4	99.4	99.3	99.2	98.8
Poly	0.4	0.5	0.6	0.6	0.7	1.1
INN	0.1	0.1	0.1	0.1	0.1	0.1

Table 24: Average weighting of individual SVM by feature class

Class	Δt					
	5	**10**	**20**	**50**	**100**	**200**
Price	54.0	56.7	57.2	59.3	62.3	59.5
Volume	31.5	33.2	33.3	32.1	29.5	30.2
Fisher	14.6	10.1	9.5	8.7	8.2	10.3

Table 25: % accuracy of benchmark, average kernel, full MKL set and Subsets A & B

Set	Δt					
	5	**10**	**20**	**50**	**100**	**200**
BM	93.0	85.6	73.4	48.8	41.6	45.7
Average	52.9	37.6	43.1	39.8	30.3	39.7
FullSet	83.4	70.6	59.3	43.8	44.6	43.8
SubsetA	79.6	68.5	59.3	46.2	45.4	44.3
SubsetB	73.7	63.9	58.3	52.2	48.5	53.4

Table 26: Stability of benchmark, average kernel, full MKL set and Subsets A & B

Set	Δt					
	5	**10**	**20**	**50**	**100**	**200**
BM	159.7	51.7	19.4	10.7	8.8	6.9
Average	25.6	18.8	19.8	21.8	21.6	14.2
FullSet	26.4	20.9	17.5	17.1	15.1	10.8
SubsetA	20.1	18.9	16.7	16.5	16.2	10.4
SubsetB	15.8	14.1	14.8	14.7	14.2	10.2

Table 27: Average weighting for Subset A

Kernel	Δt					
	5	**10**	**20**	**50**	**100**	**200**
F_1RBF_1	5.9	9.6	13.7	18.8	19.0	21.1
F_2RBF_1	30.2	31.5	28.4	25.6	24.8	24.4
F_3RBF_1	7.8	7.5	6.3	6.2	6.7	7.5
F_3RBF_2	6.6	6.7	6.1	5.8	6.7	6.5
F_3RBF_5	12.6	9.7	10.8	8.9	10.3	9.1
F_5RBF_5	19.3	16.4	17.8	17.7	17.9	16.1
F_6RBF_5	17.6	18.6	16.9	17.0	14.6	15.4

Table 28: Average weighting for Subset B

Kernel	Δt					
	5	**10**	**20**	**50**	**100**	**200**
F_1RBF_1	6.6	10.9	14.6	18.2	19.2	18.9
F_2RBF_1	14.0	13.9	12.6	12.0	11.8	11.2
F_4RBF_1	19.5	19.6	21.2	17.0	18.1	19.1
F_6RBF_1	22.7	26.4	24.4	26.1	23.8	22.4
$F_{10}RBF_1$	12.8	7.9	6.9	6.4	6.1	7.1
F_1RBF_2	0.3	0.4	1.3	2.8	2.8	3.9
F_2RBF_2	24.0	20.6	18.3	15.4	16.1	14.1
F_1RBF_5	0.1	0.3	0.7	2.1	2.2	3.3

Table 29: % accuracy comparison between kernel learning methods for full MKL set

Method	Δt					
	5	**10**	**20**	**50**	**100**	**200**
No Learning	83.4	70.6	59.3	43.8	44.6	43.8
Serial Hyper. and Fish.	83.0	71.8	59.4	46.5	47.8	45.1
Parallel Hyper. and Fish.	82.8	72.8	59.7	46.4	47.8	45.4
Serial Fish.	84.3	72.4	60.4	47.3	47.7	45.3
Parallel Fish.	84.2	73.3	60.5	46.6	47.6	45.3

Table 30: % accuracy comparison between kernel learning methods for Subset A

Method	Δt					
	5	**10**	**20**	**50**	**100**	**200**
No Learning	79.6	68.5	59.3	46.2	45.4	44.3
Serial Hyper. and Fish.	81.2	71.5	61.7	49.7	48.3	46.1
Parallel Hyper. and Fish.	81.2	71.5	61.7	49.7	48.3	46.1
Serial Fish.	82.7	72.3	61.7	49.1	48.7	45.5
Parallel Fish.	82.7	72.3	61.7	49.1	48.7	45.5

Table 31: % accuracy comparison between kernel learning methods for Subset B

Method	Δt					
	5	**10**	**20**	**50**	**100**	**200**
No Learning	73.7	63.9	58.3	52.2	48.5	53.4
Serial Hyper. and Fish.	82.1	70.8	61.3	51.2	49.6	46.5
Parallel Hyper. and Fish.	82.1	70.9	61.1	51.2	49.5	46.5
Serial Fish.	82.8	71.9	62.1	50.9	49.3	47.1
Parallel Fish.	82.8	71.9	62.1	50.9	49.3	47.1

Table 32: % β reduction from Kernel Learning for full MKL set

Method	Δt					
	5	**10**	**20**	**50**	**100**	**200**
Serial Hyper. and Fish.	2.2	2.2	3.2	3.2	3.2	3.9
Parallel Hyper. and Fish.	2.1	2.1	3.1	3.1	2.9	3.8
Serial Fish.	1.4	1.1	1.9	1.8	1.7	2.4
Parallel Fish.	1.4	1.0	1.8	1.7	1.4	2.5

Table 33: % β reduction from Kernel Learning for Subset A

Method	Δt					
	5	**10**	**20**	**50**	**100**	**200**
Serial Hyper. and Fish.	1.2	1.8	2.4	2.4	2.4	2.8
Parallel Hyper. and Fish.	1.2	1.8	2.4	2.4	2.4	2.8
Serial Fish.	0.0	0.0	0.1	0.1	0.1	0.1
Parallel Fish.	0.0	0.0	0.1	0.1	0.1	0.1

Table 34: % β reduction from Kernel Learning for Subset B

Method	Δt					
	5	**10**	**20**	**50**	**100**	**200**
Serial Hyper. and Fish.	1.1	1.7	2.0	2.1	2.5	3.0
Parallel Hyper. and Fish.	1.1	1.7	2.0	2.1	2.5	3.0
Serial Fish.	0.5	0.7	0.5	0.5	0.7	1.0
Parallel Fish.	0.5	0.7	0.5	0.5	0.7	1.0

Table 35: Correlation between % β reduction and % accuracy

Set	Δt					
	5	**10**	**20**	**50**	**100**	**200**
Full set	0.10	0.07	0.15	0.08	-0.02	-0.25
Subset A	0.04	0.06	0.14	0.04	0.03	-0.16
Subset B	-0.06	-0.04	0.02	0.00	0.12	-0.27

Table 36: Correlation between final β Value and % accuracy

Set	Δt					
	5	**10**	**20**	**50**	**100**	**200**
Full set	-0.29	-0.50	-0.38	-0.22	0.02	-0.09
Subset A	0.07	0.00	0.07	0.15	0.02	-0.07
Subset B	0.03	-0.29	-0.12	0.12	-0.07	-0.31

Table 37: Correlation between learnt and ML estimates of Fisher parameters

Method	**Parameter**							
	Serial	**Parallel**	**SerialF**	**ParallelF**	**SerialReduxB**	**ParallelReduxB**	**SerialReduxFB**	**ParallelReduxFB**
w_{Bid}	0.18	0.19	0.18	0.19	0.19	0.19	0.19	0.19
q_{Bid}	-0.65	-0.56	-0.65	-0.56	-0.65	-0.56	-0.65	-0.56
p_{Bid}	-0.03	0.00	-0.03	0.00	-0.03	-0.00	-0.03	-0.00
L_{Bid}	0.63	0.62	0.63	0.62	0.63	0.61	0.63	0.61
w_{Ask}	0.21	0.20	0.21	0.20	0.21	0.20	0.21	0.20
q_{Ask}	-0.69	-0.58	-0.69	-0.58	-0.69	-0.58	-0.69	-0.58
p_{Ask}	-0.09	-0.05	-0.09	-0.05	-0.08	-0.04	-0.08	-0.04
L_{Ask}	0.64	0.62	0.64	0.62	0.65	0.63	0.65	0.63
λ_5^{Bid}	0.53	0.56	0.53	0.56	0.53	0.55	0.53	0.55
λ_4^{Bid}	0.50	0.51	0.50	0.51	0.50	0.50	0.50	0.50
λ_3^{Bid}	0.52	0.55	0.52	0.55	0.53	0.54	0.53	0.54
λ_2^{Bid}	0.53	0.58	0.53	0.58	0.53	0.56	0.53	0.56
λ_1^{Bid}	0.51	0.56	0.51	0.56	0.51	0.56	0.51	0.56
λ_1^{Ask}	0.54	0.59	0.54	0.59	0.55	0.58	0.55	0.58
λ_2^{Ask}	0.50	0.58	0.50	0.58	0.51	0.57	0.51	0.57
λ_3^{Ask}	0.52	0.57	0.52	0.57	0.52	0.55	0.52	0.55
λ_4^{Ask}	0.40	0.44	0.40	0.44	0.40	0.42	0.40	0.42
λ_5^{Ask}	0.42	0.45	0.42	0.45	0.42	0.45	0.42	0.45
μ_{Bid}	0.06	0.10	0.06	0.10	0.07	0.11	0.07	0.11
μ_{Ask}	-0.00	0.03	-0.00	0.03	0.03	0.05	0.03	0.05
σ_{Bid}	0.51	0.41	0.51	0.41	0.51	0.40	0.51	0.40
σ_{Ask}	0.50	0.30	0.50	0.30	0.49	0.29	0.49	0.29

Table 38: Computational time of experiments

	Times (s)
Benchmark	1.64e-001
Individual	3.85e+004
Standard	8.39e+003
StandardReduxA	4.50e+003
StandardReduxB	3.10e+003
Serial	2.10e+004
Parallel	1.99e+004
SerialF	2.36e+004
ParallelF	2.53e+004
SerialReduxA	2.49e+004
ParallelReduxA	2.36e+004
SerialReduxFA	2.38e+004
ParallelReduxFA	6.03e+004
SerialReduxB	2.65e+004
ParallelReduxB	2.29e+004
SerialReduxFB	2.92e+004
ParallelReduxFB	2.22e+004

Table 39: Overall experimental accuracy comparison

Experiment	Δt					
	5	**10**	**20**	**50**	**100**	**200**
Benchmark	93.0	85.6	73.4	48.8	41.6	45.7
Average	78.1	64.5	52.7	40.7	39.3	40.8
Individual Best	90.4	82.3	71.4	49.9	47.6	51.2
Standard	83.4	70.6	59.3	43.8	44.6	43.8
StandardReduxA	79.6	68.5	59.3	46.2	45.4	44.3
StandardReduxB	73.7	63.9	58.3	52.2	48.5	53.4
Serial	83.0	71.8	59.4	46.5	47.8	45.1
Parallel	82.8	72.8	59.7	46.4	47.8	45.4
SerialF	84.3	72.4	60.4	47.3	47.7	45.3
ParallelF	84.2	73.3	60.5	46.6	47.6	45.3
SerialReduxA	81.2	71.5	61.7	49.7	48.3	46.1
ParallelReduxA	81.2	71.5	61.7	49.7	48.3	46.1
SerialReduxFA	82.7	72.3	61.7	49.1	48.7	45.5
ParallelReduxFA	82.7	72.3	61.7	49.1	48.7	45.5
SerialReduxB	82.1	70.8	61.3	51.2	49.6	46.5
ParallelReduxB	82.1	70.9	61.1	51.2	49.5	46.5
SerialReduxFB	82.8	71.9	62.1	50.9	49.3	47.1
ParallelReduxFB	82.8	71.9	62.1	50.9	49.3	47.1

Table 40: Proportion of times predictions possible for different methods

Experiment	Δt					
	5	**10**	**20**	**50**	**100**	**200**
Benchmark	61.4	70.8	79.6	86.2	85.2	70.4
Average	30.2	39.1	46.1	41.1	34.8	23.3
Individual Best	31.6	39.6	41.6	30.6	15.3	16.5
Standard	26.2	34.6	38.9	34.3	28.5	19.1
StandardReduxA	29.2	37.0	41.2	38.4	30.9	20.9
StandardReduxB	28.9	36.3	42.4	38.9	32.3	21.5
Serial	26.5	34.0	36.7	36.0	27.8	18.7
Parallel	26.4	33.8	36.7	35.6	28.7	18.5
SerialF	26.4	34.7	37.4	35.8	27.5	18.5
ParallelF	26.5	34.4	37.5	35.7	28.4	18.8
SerialReduxA	29.0	36.8	41.1	37.8	31.0	20.6
ParallelReduxA	29.0	36.8	41.1	37.8	31.0	20.6
SerialReduxFA	28.8	36.7	40.7	37.9	31.1	20.7
ParallelReduxFA	28.8	36.7	40.7	37.9	31.1	20.7
SerialReduxB	28.8	35.7	41.2	37.6	31.2	21.1
ParallelReduxB	28.8	35.7	41.2	37.6	31.2	21.1
SerialReduxFB	29.1	36.0	40.9	37.2	31.4	20.9
ParallelReduxFB	29.1	36.0	40.9	37.1	31.4	20.9

References

[1] T. Fletcher, Z. Hussain, and J. Shawe-Taylor, "Multiple kernel learning on the limit order book," in *JMLR Proceedings*, vol. 11, 2010, pp. 167–174.

[2] G. Box and G. Jenkins, *Time series analysis: Forecasting and control*. San Francisco, USA: Holden-Day, Inc., 1970.

[3] R. F. Engle, "Autoregressive conditional heteroscedasticity with estimates of variance of united kingdom inflation," *Econometrica*, vol. 50, no. 4, pp. 987–1008, 1982.

[4] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," *Journal of Econometrics*, vol. 31, no. 3, pp. 307–327, 1986.

[5] J. Hamilton, *Time Series Analysis*. USA: Princeton University Press, 1994.

[6] C. Alexander, *Market Models: A Guide to Financial Data Analysis*. UK: Wiley, 2001.

[7] A. W. Lo and A. C. Mackinlay, *A Non-Random Walk Down Wall Street*. USA: Princeton University Press, 2002.

[8] E. F. Fama, "Efficient capital markets: A review of theory and empirical work," *Journal of Finance*, vol. 25, no. 2, pp. 383–417, 1970.

[9] E. F. Fama and K. R. French, "Permanent and temporary components of stock prices," *Journal of Political Economy*, vol. 96, no. 2, p. 246, 1988.

[10] A. W. Lo and A. C. Mackinlay, "Stock market prices do not follow random walks: Evidence from a simple specification test," *Review of Financial Studies*, vol. 1, no. 1, pp. 41–66, 1988.

[11] A. W. Lo and A. C. MacKinlay, "When are contrarian profits due to stock market overreaction?" National Bureau of Economic Research, Inc, NBER Working Papers 2977, Feb. 1991.

[12] N. Jegadeesh, "Evidence of predictable behavior of security returns," *Journal of Finance*, vol. 45, no. 3, pp. 881–98, July 1990.

[13] J. Y. Campbell, A. Lo, and A. C. Mackinlay, *The Econometrics of Financial Markets*. Princeton University Press, 1996.

[14] D.-H. Ahn, J. Boudoukh, M. Richardson, and R. F. Whitelaw, "Partial adjustment or stale prices? implications from stock index and futures return autocorrelations," *Review of Financial Studies*, vol. 15, no. 2, pp. 655–689, March 2002.

[15] B. Zhou, "High-frequency data and volatility in foreign-exchange rates," *Journal of Business & Economic Statistics*, vol. 14, no. 1, pp. 45–52, January 1996.

[16] R. Cont, "Empirical properties of asset returns: stylized facts and statistical issues. quantitative finance," *Quantitative Finance*, vol. 1, pp. 223–236, 2001.

[17] M. Dacorogna, R. Gencay, U. Muller, R. Olsen, and O. Pictet, *An Introduction to High-Frequency Finance*. Academic Press, San Diego, 2001.

[18] D. Brown and R. Jennings, "On technical analysis," *Review of Financial Studies*, vol. 2, no. 4, pp. 527–551, 1989.

[19] S. N. Neftci, "Naive trading rules in financial markets and wiener-kolmogorov prediction theory: A study of technical analysis," *The Journal of Business*, vol. 64, no. 4, p. 549, 1991.

[20] M. P. Taylor and H. Allen, "The use of technical analysis in the foreign exchange market," *Journal of International Money and Finance*, vol. 11, no. 3, pp. 304–314, June 1992.

[21] Y.-H. Lui and D. Mole, "The use of fundamental and technical analyses by foreign exchange dealers: Hong kong evidence," *Journal of International Money and Finance*, vol. 17, no. 3, pp. 535–545, June 1998.

[22] W. Brock, J. Lakonishok, and B. Lebaron, "Simple technical trading rules and the stochastic properties of stock returns," Wisconsin Madison - Social Systems, Working papers 90-22, 1991.

[23] R. Dittmar, C. J. Neely, and P. Weller, "Is technical analysis in the foreign exchange market profitable? a genetic programming approach," C.E.P.R. Discussion Papers, CEPR Discussion Papers 1480, Sep. 1996.

[24] B. LeBaron, "Technical trading rule profitability and foreign exchange intervention," National Bureau of Economic Research, Inc, NBER Working Papers 5505, Mar. 1996.

[25] A. W. Lo, H. Mamaysky, and J. Wang, "Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation," National Bureau of Economic Research, Inc, NBER Working Papers 7613, 2000.

[26] "On the profitability of technical trading rules based on artificial neural networks:: Evidence from the madrid stock market," *Economics Letters*, vol. 69, no. 1, pp. 89 – 94, 2000.

[27] C. J. Neely and P. A. Weller, "Intraday technical trading in the foreign exchange market," Federal Reserve Bank of St. Louis, Working Papers 1999-016, 2001.

[28] K. A. Kavajecz, "Technical analysis and liquidity provision," *Review of Financial Studies*, vol. 17, no. 4, pp. 1043–1071, 2004.

[29] BIS, "Triennial central bank survey of foreign exchange and derivatives market activity in 2007," http://www.bis.org/publ/rpfxf07t.htm, 2007.

[30] T. Bollerslev and I. Domowitz, *Some effects of restricting the electronic order book in an automated trade execution system.* Santa Fe Institute. Boulder, CO, USA, 1993, vol. 14, p. 221252.

[31] Y. Hamao and J. Hasbrouck, "Securities trading in the absence of dealers: Trades and quotes on the tokyo stock exchange," *Review of Financial Studies*, vol. 8, no. 3, pp. 849–78, 1995.

[32] S. Maslov and M. Mills, "Price fluctuations from the order book perspective - empirical facts and a simple model," arXiv.org, Quantitative Finance Papers, Feb 2001.

[33] J.-P. Bouchaud, M. Mezard, and M. Potters, "Statistical properties of stock order books: Empirical results and models," *Quantitative Finance*, vol. 2, no. 4, pp. 251–256, 2002.

[34] I. Zovko and D. Farmer, "The power of patience: A behavioural regularity in limit-order placement," *Quantitative Finance*, vol. 2, no. 5, pp. 387–392, 2002.

[35] M. Potters and J.-P. Bouchaud, "More statistical properties of order books and price impact," Science & Finance, Capital Fund Management, Science & Finance (CFM) working paper archive 0210710, Oct. 2002.

[36] N. Hautsch and T. Hall, "A continuous-time measurement of the buy-sell pressure in a limit order book market," *SSRN eLibrary*, 2004.

[37] J.-P. Bouchaud, Y. Gefen, M. Potters, and M. Wyart, "Fluctuations and response in financial markets: the subtle nature of 'random' price changes," *Quantitative Finance*, vol. 4, no. 2, pp. 176–190, 2004.

[38] P. Weber and B. Rosenow, "Order book approach to price impact," *Quantitative Finance*, vol. 5, no. 4, pp. 357–364, 2005.

[39] M. Garman, "Market microstructure," *Journal of Financial Economics*, vol. 3, no. 3, pp. 257–275, 1976.

[40] D. Morse and N. Ushman, "The effect of information announcements on the market microstructure," *The Accounting Review*, vol. 58, no. 2, pp. 247–258, 1983.

[41] R. Lease, R. Masulis, and J. Page, "An investigation of market microstructure impacts on event study returns," *Journal of Finance*, vol. 46, no. 4, pp. 1523–36, September 1991.

[42] F. Allen and G. Gorton, "Stock price manipulation, market microstructure and asymmetric information," National Bureau of Economic Research, Inc, NBER Working Papers 3862, Oct. 1991. [Online]. Available: http://ideas.repec.org/p/nbr/nberwo/3862.html

[43] R. Huang and H. Stoll, "Market microstructure and stock return predictions," *Review of Financial Studies*, vol. 7, no. 1, pp. 179–213, 1994.

[44] B. Brennan and A. Subrahmanyam, "Market microstructure and asset pricing: On the compensation for illiquidity in stock returns," *Journal of Financial Economics*, vol. 41, no. 3, pp. 441–464, July 1996.

[45] Y. Amihud, H. Mendelson, and B. Lauterbach, "Market microstructure and securities values: Evidence from the tel aviv stock exchange," New York University, Leonard N. Stern School of Business-, New York University, Leonard N. Stern School Finance Department Working Paper Seires 98-004, Oct. 1997.

[46] G. MacKinnon and H. Nemiroff, "Liquidity and tick size: Does decimalization matter?" *Journal of Financial Research*, vol. 22, no. 3, pp. 287–99, Fall 1999.

[47] M. O'Hara, *Market Microstructure Theory*. Blackwell Publishing Ltd, 1995.

[48] J. Hasbrouck, *Modeling market microstructure time series*, 1996, ch. 22, pp. 647–692.

[49] A. Lo, A. MacKinlay, and J. Zhang, "Econometric Models of Limit-Order Executions," *SSRN eLibrary*, 1997.

[50] A. Madhavan, "Market microstructure: A survey," *Journal of Financial Markets*, vol. 3, no. 3, pp. 205–258, August 2000.

[51] L. Harris, *Trading and Exchanges: Market Microstructure for Practitioners*. Oxford University Press, 2002.

[52] J. Hasbrouk, *Empirical Market Microstructure: The Institutions, Economics, and Econometrics of Securities Trading*. Oxford University Press, USA, 2006.

[53] C. Parlour and D. Seppi, "Limit order markets: A survey," in *Handbook of Financial Intermediation and Banking*, A. Boot and A. Thakor, Eds. Amsterdam, The Netherlands: Elsevier Science, 2008, ch. 3, pp. 61–96.

[54] E. Jondeau, A. Perilla, and G. M. Rockinger, "Optimal Liquidation Strategies in Illiquid Markets," *SSRN eLibrary*, 2008.

[55] J. Linnainmaa and I. Rosu, "Time Series Determinants of Liquidity in a Limit Order Market," *SSRN eLibrary*, 2008.

[56] C. Neely, P. Weller, and R. Dittmar, "Is technical analysis in the foreign exchange market profitable?: A genetic programming approach," *Journal of Financial and Quantitative Analysis*, vol. 32, no. 4, pp. 405–426, 1997.

[57] W. Banzhaf, J. Koza, C. Ryan, L. Spector, and C. Jacob, "Genetic programming," *Intelligent Systems and their Applications*, vol. 15, no. 3, pp. 74–84, 2000.

[58] F. Allen and R. Karjalainen, "Using genetic algorithms to find technical trading rules," *Journal of Financial Economics*, vol. 51, no. 2, pp. 245–271, 1999.

[59] D. Goldberg, Ed., *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Kluwer Academic Publishers, 1989.

[60] M. A. Kaboudan, "Genetic programming prediction of stock prices," *Computational Economics*, vol. 16, no. 3, pp. 207–236, 2000.

[61] J. Potvin, S. P., and M. Vallee, "Generating trading rules on the stock markets with genetic programming," *Computers and Operations Research*, vol. 31, no. 7, pp. 1033–1047, 2004.

[62] C. Park and S. Irwin, "The profitability of technical analysis: A review," Master's thesis, University of Illinois, Urbana-Champaign, Urbana, 2004.

[63] T. Fletcher, "Hybrid evolutionary techniques for fx arbitrage prediction," Master's thesis, University of Sussex, UK, 2007.

[64] K. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[65] R. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. D, pp. 35–45, 1960.

[66] S. Julier and J. Uhlmann, "A new extension of the kalman filter to nonlinear systems," *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, 1997.

[67] G. Evensen, "The ensemble kalman filter: Theoretical formulation and practical implementation," *Ocean Dynamics*, vol. 53, pp. 343–367, 2003.

[68] *A robust non-linear multivariate Kalman filter for arbitrage identification in high frequency data*, ser. Neural Networks in Financial Engineering: Proceedings of the NNCM-95, 1996.

[69] P. J. Bolland and J. Connor, "A constrained neural network kalman filter for price estimation in high frequency financial data," *International Journal of Neural Systems*, vol. 8, no. 4, pp. 399–415, 1997.

[70] P. Idvall and C. Jonsson, "Algorithmic trading: Hidden markov models on foreign exchange data," Master's thesis, Linkopings Universitet, Sweden, 2008.

[71] R. Hassan, B. Nath, and M. Kirley, "A fusion model of hmm, ann and ga for stock market forcasting," *Expert Systems with Applications*, vol. 33, pp. 171–180, 2007.

[72] R. Hassan and B. Nath, "Stock market forecasting using hidden markov model: A new approach," *Proceedings of the 2005 5th International Conference on Intelligent Systems Design and Applications*, 2005.

[73] C. Landen, "Bond pricing in a hidden markov model of the short rate," *Finance and Stochastics*, vol. 4, pp. 371–389, 2000.

[74] R. Mamon and R. Elliott, Eds., *Hidden Markov Models in Finance*. Springer Science Business Media, 2007.

[75] Z. Yingjian, "Prediction of financial time series with hidden markov models," Master's thesis, Shandong University, China, 2004.

[76] J. Bulla, "Application of hidden markov models and hidden semi-markov models to financial time series," Master's thesis, Georg-August-Universitat Gottingen, Germany, 2006.

[77] A. Rossi and G. Gallo, "Volatility estimation via hidden markov models," *Journal of Empirical Finance*, vol. 13, pp. 203–230, 2006.

[78] T. Ryden, T. Terasvirta, and S. Asbrinkc, "Stylised facts of daily return series and the hidden markv model," *Journal of Applied Econometrics*, vol. 13, pp. 217–244, 1998.

[79] R. R. Trippi and E. Turban, Eds., *Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real World Performance*. New York, NY, USA: McGraw-Hill, Inc., 1992.

[80] C.-M. Kuan and T. Liu, "Forecasting exchange rates using feedforward and recurrent neural networks," *Journal of Applied Econometrics*, vol. 10, no. 4, pp. 347–64, Oct.-Dec. 1995.

[81] S. Walczak, "An empirical analysis of data requirements for financial forecasting with neural networks," *J. Manage. Inf. Syst.*, vol. 17, no. 4, pp. 203–222, 2001.

[82] J. Shadbolt and J. G. Taylor, Eds., *Neural networks and the financial markets: predicting, combining and portfolio optimisation*. London, UK: Springer-Verlag, 2002.

[83] C. Cortes and V. Vapnik, "Support vector networks," in *Machine Learning*, 1995, pp. 273–297.

[84] F. Tay and L. Cao, "Application of support vector machines in financial time series forecasting," *Omega*, vol. 29, pp. 309–317, 2001.

[85] ——, "Modified support vector machines in financial time series forecasting," *Neurocomputing*, vol. 48, pp. 847–861, 2002.

[86] L. Cao, "Support vector machines experts for time series forecasting," *Neurocomputing*, vol. 51, pp. 321–339, 2003.

[87] K. Kim, "Financial time series forecasting using support vector machines," *Neurocomputing*, vol. 55, pp. 307–319, 2003.

[88] F. Perez-cruz, J. Afonso-rodriguez, and J. Giner, "Estimating garch models using support vector machines," *Quantitative Finance*, vol. 3, no. 3, pp. 163–172, 2003.

[89] W. Huang, Y. Nakamori, and S.-Y. Wang, "Forecasting stock market movement direction with support vector machine," *Comput. Oper. Res.*, vol. 32, no. 10, pp. 2513–2522, 2005.

[90] T. V. Gestel, J. A. K. Suykens, D.-E. Baestaens, A. Lambrechts, G. Lanckriet, B. Vandaele, B. D. Moor, and J. Vandewalle, "Financial time series prediction using least squares support vector machines within the evidence framework," in *IEEE Transactions on Neural Networks*, 2001, pp. 809–821.

[91] N. Hazarika and J. G. Taylor, *Predicting bonds using the linear relevance vector machine.* Springer-Verlag, 2002, ch. 17, pp. 145–155.

[92] N. N. P. Tino and X. Yao, "Volatility forecasting with sparse bayesian kernel models," in *Proc. 4th International Conference on Computational Intelligence in Economics and Finance, Salt Lake City, UT*, 2005, pp. 1150–1153.

[93] S.-C. Huang and T.-K. Wu, "Wavelet-based relevance vector machines for stock index forecasting," in *International Joint Conference on Neural Networks (IJCNN)*, 2006, pp. 603–609.

[94] ——, "Combining wavelet-based feature extractions with relevance vector machines for stock index forecasting," *Expert Systems*, vol. 25, pp. 133–149, 2008.

[95] T. Fletcher, F. Redpath, and J. D'Alessandro, "Machine learning in fx carry basket prediction," in *Proceedings of the International Conference of Financial Engineering*, vol. 2, 2009, pp. 1371–1375.

[96] S. K. Chalup and A. Mitschele, "Kernel methods in finance," in *Handbook on Information Technology in Finance*, 2008, pp. 655–687.

[97] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *J. Mach. Learn. Res.*, vol. 1, pp. 211–244, 2001.

[98] D. J. C. Mackay, "Bayesian interpolation," *Neural Computation*, vol. 4, pp. 415–447, 1992.

[99] L. Blume, D. Easley, and M. O'Hara, "Market statistics and technical analysis: The role of volume," *Journal of Finance*, vol. 49, no. 1, pp. 153–81, March 1994.

[100] C. Lee, B. Swaminathan, R. S. (editor, A. Subrahmanyam, and Y. Soejima, "Price momentum and trading volume," *Journal of Finance*, vol. 55, pp. 2017–2069, 1998.

[101] C. Marney, "Building robust fx trading systems," *FX Trader Magazine*, 2010.

[102] C.-H. Park and S. H. Irwin, "The profitability of technical analysis: A review," University of Illinois at Urbana-Champaign, Department of Agricultural and Consumer Economics, AgMAS Project Research Reports 37487, 2004.

[103] C. J. Neely, P. A. Weller, and J. M. Ulrich, "The adaptive markets hypothesis: Evidence from the foreign exchange market," *Journal of Financial and Quantitative Analysis*, vol. 44, no. 02, pp. 467–488, April 2009.

[104] Y. Ephraim and W. J. J. Roberts, "Revisiting autoregressive hidden markov modeling of speech signals," *IEEE Signal Processing Letters*, vol. 12, pp. 166–169, 2005.

[105] P. Fearnhead, "Exact bayesian curve fitting and signal segmentation," *IEEE Transactions on Signal Processsing*, vol. 53, pp. 2160–2166, 2005.

[106] ——, "Exact and efficient bayesian inference for multiple changepoint problems," *Statistics and Computing*, vol. 16, pp. 203–213, June 2006.

[107] C.-M. Kuan and T. Liu, "Forecasting exchange rates using feedforward and recurrent neural networks," *Journal of Applied Econometrics*, vol. 10, pp. 347–364, 1995.

[108] G. R. G. Lanckriet, T. De Bie, N. Cristianini, M. I. Jordan, and W. S. Noble, "A statistical framework for genomic data fusion," *Bioinformatics*, vol. 20, no. 16, pp. 2626–2635, 2004.

[109] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan, "Multiple kernel learning, conic duality, and the smo algorithm," in *ICML '04: Proceedings of the twenty-first international conference on Machine learning*. New York, NY, USA: ACM, 2004, p. 6.

[110] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artificial Intelligence*, vol. 97, no. 1-2, pp. 245 – 271, 1997.

[111] L. Wang and J. Zhu, "Financial market forecasting using a two-step kernel learning method for the support vector regression," *Annals of Operation Research*, vol. 174, pp. 103–120, 2010.

[112] R. Luss and A. d'Aspremont, "Predicting abnormal returns from news using text classification," 2009.

[113] C. Ullrich, D. Seese, and S. Chalup, "Foreign exchange trading with support vector machines," in *Advances in Data Analysis*. Springer Berlin Heidelberg, 2007, pp. 539–546.

[114] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "Simplemkl," *Journal of Machine Learning Research*, vol. 9, pp. 2491–2521, November 2008.

[115] Z. Hussain and Shawe-Taylor, "Metric learning analysis," PinView FP7-216529 Project Deliverable Report D3.2, December 2009. [Online]. Available: http://www.pinview.eu/deliverables.php

[116] R. E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, no. 2, pp. 197–227, 1990.

[117] A. Demiriz, K. P. Bennett, and J. Shawe-Taylor, "Linear programming boosting via column generation," *Machine Learning*, vol. 46(13), pp. 225–254, 2002.

[118] G. Appel, *Technical Analysis: Power Tools for Active Investors*. Financial Times, 2005.

[119] C. Ullrich, *Forecasting and Hedging in the Foreign Exchange Markets*. Springer, 2009.

[120] C. Williams, "Computation with infinite neural networks," *Neural Computation*, vol. 10, no. 5, pp. 1203–1216, 1998.

[121] T. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *In Advances in Neural Information Processing Systems 11*. MIT Press, 1998, pp. 487–493.

[122] D. Easley and M. O'Hara, "Time and the process of security price adjustment," *Journal of Finance*, vol. 47, no. 2, pp. 576–605, June 1992.

[123] R. Engle and J. Russell, "Autoregressive conditional duration: A new model for irregularly spaced transaction data," *Econometrica*, vol. 66, pp. 1127–1162, 1998.

[124] D. Easley, R. F. Engle, M. O'Hara, and L. Wu, "Time-varying arrival rates of informed and uninformed trades," EconWPA, Tech. Rep., 2002.

[125] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the nelder–mead simplex method in low dimensions," *SIAM J. on Optimization*, vol. 9, pp. 112–147, May 1998. [Online]. Available: http://dx.doi.org/10.1137/S1052623496303470

[126] T. Rydberg and N. Shephard, *A modelling framework for the prices and times of trades made on the New York stock exchange*. Cambridge University Press, 2000.

[127] F. Mainardi, M. Raberto, R. Gorenflo, and E. Scalas, "Fractional calculus and continuous-time finance ii: the waiting- time distribution," *Physica A: Statistical Mechanics and its Applications*, vol. 287, p. 468, 2000.

[128] F. Mainardi, R. Gorenflo, and E. Scalas, "A fractional generalization of the poisson processes," *Vietnam Journal of Mathematics*, vol. 32, pp. 53–64, 2004.

[129] M. Politi and E. Scalas, "Activity spectrum from waiting-time distribution," *Physica A: Statistical Mechanics and its Applications*, vol. 383, no. 1, p. 43, 2007.

[130] ——, "Fitting the empirical distribution of intertrade durations," *Physica A: Statistical Mechanics and its Applications*, vol. 387, no. 8-9, p. 2025, 2008.

[131] T. Lancaster, *The Econometric Analysis of Transition Data*. Cambridge University Press, 1992.

[132] D. Barber, *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2011, in press.

[133] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances in Large Margin Classifiers*, 1999, pp. 61–74.

[134] C. Faith, *Way of the Turtle*. McGraw-Hill Professional, 2007.

[135] J. A. Bollinger, *Bollinger on Bollinger Bands*. McGraw-Hill, 2001.

[136] J. W. Wilder, *New Concepts in Technical Trading Systems*. Trend Research, 1978.