Forecasting Financial Time Series with Multiple Kernel Learning

Luis Fábregues, Argimiro Arratia, and Lluís A. Belanche

Department of Computer Science**
Universitat Politècnica de Catalunya, Jordi Girona, 1-3 08034, Barcelona, Spain
luis.fabregues@est.fib.upc.edu,argimiro@cs.upc.edu,belanche@cs.upc.edu

Abstract. This paper introduces a forecasting procedure based on multivariate dynamic kernels to re-examine –under a non linear framework– the experimental tests reported by Welch and Goyal showing that several variables proposed in the academic literature are of no use to predict the equity premium under linear regressions. For this approach kernel functions for time series are used with multiple kernel learning in order to represent the relative importance of each of these variables.

Keywords: Forecasting · Support vector classification · Financial time series · Multiple kernel learning · Time series kernels

1 Introduction

There is a long history of attempts to predict stock market returns by specifying some regression on lagged predictor variables independent of the stock market returns. Shiller [20], Campbell and Shiller [5], Cochrane [9], among others, have studied the forecasting of future excess returns using the dividend price ratio as predictor. Other popular predictor variables explored in the literature are the dividend yield, earnings price ratio, dividend-to-earnings ratio, volatility, interest rates, exchange rates, consumption indices and inflation rates (see, e.g., [15], [16], [17], [18] and [11] for a general discussion). The list of valuation ratios sought of as forecasters of expected excess returns is much longer and show *"... a pervasive pattern of predictability across markets wherein the cashflow or price change one may have expected is not what is forecast."* [11]. In view of this and further evidence showing the spurious nature of predictor models (mostly linear regressions on the aforementioned valuation ratios), several authors have conducted extensive studies on the forecasting performance of various economic variables and different models (to mention a few, e.g., [2], [6], [10], [21]). The work by Welch and Goyal [21] is of particular interest since the authors do a comprehensive revision of the empirical performance of the most widely accepted variables as predictors of equity premium, under *linear* regression models, and

** Supported by MINECO project APCOM (TIN2014-57226-P) and *Generalitat de Catalunya* 2014 SGR 890 (MACDA).

conclude that these models have poor predictive capacity both in-sample and out-of-sample.

In this work we extend the stock return predictability test of Goyal and Welch to non-linear (and semi-parametric) classification models on the different valuation ratios. We are thus considering the possible non-linear relationship between the stock returns and the predictor variables and expanding the framework of predictability from linear to more complex models. The models we consider come from statistical learning adapted to the objective of time series forecasting. We use ν-Support Vector Machines for classification [7] with dynamic kernel functions able to analyze multivariate temporal structures, and Multiple Kernel Learning [3] to integrate different financial information and different kernels.

2 Multivariate Dynamic Kernels for Time Series

Kernels are two-place symmetric functions that evaluate an inner product of the arguments in some feature space, thereby inducing an implicit mapping that creates an image of the input into the desired feature space. It is then possible to compare input data in a higher-dimensional space without the need of calculating the exact coordinates of the transformation. Kernel functions can be defined as $k(\boldsymbol{x}, \boldsymbol{z}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \rangle$, where $\boldsymbol{x}, \boldsymbol{z} \in X$ are input vectors. ϕ represents the mapping from the original feature space X into a new feature space F as $\phi : \boldsymbol{x} \rightarrow \phi(\boldsymbol{x}) \in F$. It is known that a function is a valid kernel function if and only if it induces positive semi-definite (p.s.d.) matrices $K = [k(\boldsymbol{x}_i, \boldsymbol{x}_j)]$. This property can be defined as $\boldsymbol{c}^\top K \boldsymbol{c} \geq 0$, for all $N \in \mathbb{N}, \boldsymbol{x}_1, ..., \boldsymbol{x}_N \in X$ and $\boldsymbol{c} \in \mathbb{R}^N$.

Kernels for time series can be constructed using two approaches: structural similarity and model similarity. Structural similarity employs methods to find an alignment of the data that makes possible the comparison between series. Model similarity changes the structure of the data by constructing a higher level representation thereof and the comparison is performed using this new representation.

2.1 Vector Auto-Regression Kernel

The Vector Auto-Regression (VAR) model relates the data at the observation at point $x(t)$ with a linear combination of lagged values of the observation. In order to fit a model, a lag parameter is provided, which defines how many time steps the function will be looking at in the past to assess the linear combination parameters. The formulation is as follows: $x(t) = \sum_{l=1}^{L} A_l x(t-l) + b + \varepsilon_t$, where $x(t)$ is the sample at time t, L is the number of lags of the model, A is the transition matrix (a square matrix with as many dimensions as features the data has), b is the intercept (a vector of the dimension equal to the number of features) and ε_t is the Gaussian noise at time t.

The VAR function can be used to build a model similarity kernel. In order to compare VAR models one considers the binding of transition matrices and the intercept vector as an additional column. This results in $\hat{B} = \Big[A_1|A_2|...|A_L|[b]\Big]$. The distance between series s_1 and s_2 is the Frobenius norm of the difference between $\hat{B}_{s_1}$ and $\hat{B}_{s_2}$:

$$\mathrm{FD}(s_1, s_2) = \sqrt{Trace\left\{(\hat{B}_{s_1} - \hat{B}_{s_2})(\hat{B}_{s_1} - \hat{B}_{s_2})^\top\right\}}. \tag{1}$$

Once this Frobenius distance is calculated, the distance can be transformed into a valid kernel using the Radial Basis Function (RBF) transform:

$$k_{\mathrm{VAR}} = \exp\left\{\frac{-FD(s_1, s_2)}{2\sigma}\right\} \tag{2}$$

The parameters of this kernel methodology are the number of lags L and σ. The value of L will be fixed to 5, whereas σ will be set to the median Frobenius distance between the time series being compared. Both parameters are set following [13] and our own previous experience [19].

2.2 Global Alignment Kernel

Global Alignment(GA) [12] is a generalization of a family of distance and similarities called Dynamic Time Warping (DTW), the goal of which is to measure the distance between two series. In order to do so, both series should be aligned. The core of the problem is how to determine the best alignment between the two series and, using that alignment, to measure their similarity.

An *alignment* in DTW is represented by a set of relationships between a point in the series and another point of the same series (or the other one). Considering s_1 and s_2 as two time series, those relationships are the following: $s_1(t)$ with $s_2(t)$ denoted by $\rightarrow$, $s_1(t)$ with $s_1(t+1)$ denoted by $\uparrow$ and $s_1(t)$ with $s_2(t+1)$ denoted by $\nearrow$. The relationships are represented as two integer vectors π_1, π_2 of the same length with binary increases. The length of these vectors is always equal or less to the length of the smallest series. Each relationship can be represented on those vectors as follows: $(0,1)$ for $\rightarrow$, $(1,0)$ for $\uparrow$ and $(1,1)$ for $\nearrow$. Intuitively, each vector $\pi_1(t)$ indicates an element of s_1 that forms a relationship with the element $\pi_2(t)$ of s_2. For the sake of simplicity the two vectors that represent the alignment will be denoted as π. Those alignments, by definition, only consider values of zero or one lag in both series.

After obtaining a satisfying alignment, the distance between the series can be obtained as: $D_\pi = \sum_{i=1}^{|\pi|} d(x_{\pi_1(i)}, y_{\pi_2(i)})$, where the distance function d can be any metric, most often the Euclidean distance. The presented algorithm is capable of finding more than one alignment. The selected alignment will be the one that minimizes the distance between the series. The formulation of the final MDTW distance is:

$$\mathrm{MDTW}(s_1, s_2) = \frac{1}{|\pi^*|} \min_{\pi \in A(s_1, s_2)} D_\pi(s_1, s_2), \tag{3}$$

where $|\pi^*|$ is the length of the alignment with less distance and A is the set of all possible alignments.

This distance measurement does not fulfill the p.s.d.-ness requirement to form a kernel, even after applying the RBF transformation. For this reason the GA generalization is applied, which delivers correct kernels and enables the creation of a structural similarity kernel. This follows the same computational steps as DTW; however, instead of selecting the alignment with minimum distance, it considers all the alignments. This makes kernels defined by this metric p.s.d. under mild assumptions, based in the notion that all alignments provide information about the similarities between both series.

The formulation of this kernel function can be expressed using several distance metrics. The following formula is the one used in the context of this work:

$$k_{\mathrm{GA}}(s_1, s_2) = \sum_{\pi \in A(s_1, s_2)} \exp(-D_\pi(s_1, s_2)) \tag{4}$$

An improvement over GA was introduced under the name of Fast Global Alignment or Triangular Global Alignment [14]. This improved version aims at reducing the computational time of the procedure; this is accomplished by using an extra parameter T that restricts the number of alignments taken into account during the final calculation of k_{GA}. In particular, lower values of T make the kernel function use alignments close to the diagonal. Increasing the value of T increases the range of alignments that are taken into account.

2.3 Multivariate Dynamic Euclidean Distance Kernel

In the same lines as Global Alignment Kernel, Multivariate Dynamic Euclidean Distance Kernel (MDED) is a structural similarity model that creates an alignment of data between two series of different size in order to be able to compute the distance measure. MDED opts for a much simpler approach, as it removes the first elements of the longest series until it matches the size of the shortest time series. Even if this alignment is potentially worse in most of the cases with respect to MDTW, MDED is computationally less expensive. The approach is also backed by financial theory: observations generated in later time stamps contains information from older ones. Having that $L_1 \leqslant L_2$ where L_1 and L_2 are the lengths of vectors s_1 and s_2 respectively, we define this alignment in the notation of DTW as $\pi_1 = [0, 1, 2, ..., L_1 - 1, L_1]$ and $\pi_2 = [L_2 - (L_1 - 1), L_2 - (L_1 - 2), ..., L_2 - 1, L_2]$. Using this alignment, the distance between the series is computed as in eq. (5). Again, the metric employed is the Euclidean distance. In a similar line to the k_{VAR} calculation, it is necessary to calculate the RBF kernel using the defined dissimilarity measure in order to obtain a p.s.d. kernel:

$$k_{\mathrm{MDED}} = \exp\left\{\frac{-D_\pi(s_1, s_2)}{2\sigma}\right\} \tag{5}$$

The σ parameter of this kernel function is estimated following the author's past experience [19], using the median of all D_π of the available data.

2.4 Multivariate Dynamic Arc-Cosine Kernel

Arc-Cosine kernels have several interesting properties, because their construction is related to neural networks with an infinite hidden layer [8]. A structural similarity model can be also built using this formulation. The kernel as used here depends on the angle between the series, given by $\theta = \cos^{-1}\left(\frac{s_1^\top s_2}{\|s_1\|\|s_2\|}\right)$.

The full formulation of this kernel function introduces a "degree" n, which regulates its complexity, as follows:

$$k_n(s_1, s_2) = \frac{1}{\pi}\|s_1\|^n\|s_2\|^n J_n(\theta) \tag{6}$$

where J_n is a family of functions that depend on the chosen degree and the current angle; in particular, $J_0(\theta) = \pi - \theta$ and $J_1(\theta) = \sin(\theta) + (\pi - \theta)\cos(\theta)$.

2.5 Multiple Kernel Learning

Multiple Kernel Learning (MKL) [3] aims at finding the best combination of kernels to solve a task. It is possible for a problem to have several kernel functions that cover different characteristics of the data, or different representations of the same data. Those procedures create different kernel matrices that can be used to train a predictive model. Although it is possible to combine the information of those matrices into a single combined kernel matrix, MKL makes possible to combine in the same predictive model information obtained using different techniques. The mathematical formulation of this process is the following:

$$k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = f\Big(\left\{k_m(x_i^m, x_j^m)\right\}_{m=1}^P \; ; \eta\Big), \tag{7}$$

where k_η represents the combined kernel, f_η is the (linear or non-linear) combination function, k_m represents the m-th kernel function and η parametrizes the combination. In this paper, the MKL algorithm of choice is EasyMKL [1], which obtains the parameters η of the combination function using an optimization approach. Specifically, a max-min problem is solved involving the η parameters and the probability distribution γ of each class. After the η weights are obtained they are combined convexly (the optimization restrictions ensuring that the weights are positive and sum to one). The L_1 norm is used as a structural risk function to guide the process. As the base learner EasyMKL uses KOMD, a kernel classifier that performs direct optimization of the margin distribution.

3 A non-linear predictability framework

As stated in the introduction, the objective of this work is to replicate the experiments performed by Welch and Goyal in [21] with kernel functions and using multiple kernel learning to weight each feature in order to determine its relative influence in the prediction of the equity premium. One notable difference is that this experiment will consider as output variable not the equity premium but its sign, or direction. A positive value indicates a good precondition to hold a share and a negative value indicates a proper time to sell the stock shares. This paradigm shift implies that the problem is no longer a regression one but one of classification.

The experiment uses the same data as in [21] for a partial set of the variables considered by the authors. It is a data set that comprises several financial variables measured monthly, quarterly and yearly in the range of years between 1871 and 2005, and compiled from several sources[1]. The considered variables are described below. The experimental process follows two phases: the evaluation of each combination of kernels using only endogenous variables and the comparison between exogenous variables using multiple kernel learning.

3.1 Considered Variables

Equity Premium. A representation of the stock market returns. It is calculated combining several variables from the original data $ep_t = \log((Index_t + D12_t)/Index_{t-1}) - \log(Rfree_t + 1)$ where $Rfree$ is the risk-free rate. This variable is theoretically the return rate of an investment with zero risk, however in practice it is obtained from the interest rate of a three month U.S. Treasury bill. The variables associated with $Index$ and $D12$ are the S&P 500 index and the dividends, respectively, and will be discussed below.

Stock Returns. The original problem uses S&P 500 index returns (denoted as SPX or simply $Index$), obtained from Center for Research in Security Press (CRSP) and the website of R. Shiller. This variable encapsulates the largest market capitalizations of public companies serving as an indicator of the U.S. economy and will be considered an endogenous variable.

Dividend Price Ratio. Both this variable and the next are dependent on the dividends ($D12$), which are a moving sum with a window of 12 months of the dividends paid on the S&P 500 index. The dividends data is obtained from the Shiller's website and the S&P Corporation. The formula for Dividend Price Ratio is $dp_t = \log(D12_t) - \log(Index_t)$.

Dividend Yield. Very similar to Dividend Price, but Dividend Yield considers past values of SPX $dy_t = \log(D12_t) - \log(Index_{t-1})$.

Earning Price. In a similar line to the variables created using dividends, earning price($E12$) is the moving sum of earnings from S&P 500 index in a window of 12 months. Part of this data is extracted from Shiller's website and the other part is the result from an interpolation process by the authors. The Earning Price is formulated as $ep = \log(E12_t) - \log(Index_t)$.

[1] Publicly available from `http://www.hec.unil.ch/agoyal/`.

Stock Variance. The sum of squared daily returns of SPX. This data was obtained by Welch and Goyal with the help of G. William Schwert and CRSP.

3.2 Experimental Methodology

The data has to be grouped in blocks of a given size as the kernel functions of VAR and GA need a sequence of events to extract their similarity. A simple approach is to group monthly data into yearly blocks, creating data structures that contain 12 months.

The experimental approach will be a moving window of a certain number of years. A predictive model will be validated and built with the size of the window. The output variable is the sign of the equity premium in the first month of the next year, so it becomes a classification problem. The input variables are calculated as stated in the previous section. It is included in the data frames lagged values of each feature in order to increase the information of each data entry and help the models form better temporal dependencies.

Our approach is then divided in two steps. The first is to determine which is the best kernel function or combination of them to perform the described predictive task. The second is to use the selected method to determine the relative importance of each exogenous variable using the weights of the trained Multiple Kernel Learning (MKL) model.

The first step is in accordance with the classical methodology to evaluate several models: perform a predictive task using the same data and compare the results using a predefined metric. Being a classification problem, the easiest way to compare models is to measure the accuracy of training, validation and testing. In this step, the different kernel functions considered will be used both individually and in conjunction using MKL. Each of these models will have its parameters ν, σ and λ individually tuned.

The implementation is able to use two types of validation: Out Of Sample (OOS) and Cross-Validation (CV). The inclusion of two validation procedures is due to the somewhat controversial use of CV for time series. By the nature of this approach the folds do not respect any particular order (*e.g.*, the validation part could be the oldest, using a model trained with future data). This fact often prevents the use of CV in time-dependent datasets, as it is considered 'unrealistic'. However, as it has been recently shown, CV *can* be used with stationary time series in which the predictor values are lagged versions of the response value [4].

In the case of the single kernel functions, the creation and evaluation of the model is straightforward: a support vector machine is fitted using the training data and tested or validated with the rest of the data. In the case of Multiple Kernel Learning, all kernel functions introduced in section 2 will create a kernel matrix that will be used in the MKL procedure. However, the implementations of EasyMKL do not admit any parameter, reducing the adaptability of the model. To correct this, the training procedure will fit an EasyMKL model, obtain the weights, build the combined matrix of kernels and then train a standard ν-SVM.

The second step tries to determine the relative importance of each exogenous variable by using the MKL weights. To this end, one data frame for each exogenous variable is created, each containing the exogenous variable and SPX, with different number of lags. In the list of data frames for the MKL are also included the matrix with the endogenous variable, SPX, and a data frame with all the variables. Each data frame will serve as training and validation data for the methodology selected in the first step, creating a kernel matrix that will be included in the MKL list. At the end, the performance of the models can be measured using their accuracy and the distribution of MKL weights.

Using this representation, the input to each experiment will be a list containing six matrices encapsulating the following features sets:

1. S&P 500 Index;
2. S&P 500 Index and Dividend-Price Ratio;
3. S&P 500 Index and Earning Price;
4. S&P 500 Index and Dividend Yield;
5. S&P 500 Index and Stock Variance;
6. All the features

The experiments will be considered including lagged versions of these features as additional information for the modeling process.

3.3 Performance Metrics

The performance metric that was selected for this task is the accuracy, defined as the number of correctly classified samples over the total number of samples. Three different accuracies will be taken into account in the results: train accuracy, validation accuracy and test accuracy. A second metric is given by the weights of the multiple kernel learning model. As stated in the definition of the work, the weights of a multiple kernel learning can be helpful to determine the relative importance of each kernel. This metric will help to determine which kernel function creates the best models from the data and which kernel matrix, constructed with the different variables, has a higher impact on the prediction of the results.

3.4 Empirical Results

Table 1 shows the evaluation of the different kernel methods and multiple kernel learning using EasyMKL. These results are obtained using the same data for each method but adjusting the parameters individually. All the methods are also compared using out of sample validation and cross-validation.

The results of this table show several interesting points. All the test accuracies fall within the range from 55% to 70%, which indicates the general capacity of Multivariate Dynamic kernels for this task. Individual kernels share similar test accuracies, around the 64%, including very simple kernels like k_{MDED} and k_{MDARC0}. It is also worth to mention that the kernels based on the arc-cosine

Out Of Sample Validation	k_{VAR}	k_{GA}	k_{MDED}	k_{MDARC0}	k_{MDARC1}	MKL	MKL Norm
Train Acc.	0.588	0.739	0.722	0.674	0.734	0.982	0.777
Validation Acc.	0.872	0.859	0.74	0.491	0.452	0.529	0.763
Test Acc.	0.631	0.64	0.64	0.64	0.658	0.64	0.694

Cross-Validation	k_{VAR}	k_{GA}	k_{MDED}	k_{MDARC0}	k_{MDARC1}	MKL	MKL Norm
Train Acc.	0.61	0.762	0.742	0.675	0.729	1.000	0.898
Validation Acc.	0.751	0.691	0.674	0.559	0.445	0.417	0.568
Test Acc.	0.568	0.649	0.631	0.640	0.640	0.667	0.658

Table 1. Results of kernel combinations.

kernel perform as well if not better than other more common kernel functions. In particular, k_{MDARC1} is the best performing individual kernel, surpassing Global Alignment in the version that uses out-of-sample validation.

By a considerable margin, the best performing method is the combination of kernels created with EasyMKL, surpassing all the individual kernels. This technique tends to over-fit, as it can be observed in the difference of accuracy between training, validation and testing. It is most prominent in the case of cross-validation, were training accuracy is much higher than test and validation accuracies. The normalization also has interesting repercussions on the results: it reduces training accuracy and increases validation accuracy, reducing over-fitting. The results of the normalization technique seem to differ depending of the validation technique employed.

Finally, it is also worth to comment the application of the different validation techniques on the results. There is no clear pattern to determine if cross-validation is better or worse than out-of-sample since all kernels react different. In the cases of k_{GA} and not-normalized MKL the results improve, but the rest of kernel functions have the same or worst tested accuracy. The results clearly differ with [4], specially in the case of k_{VAR}.

What follows is a table containing the resulting weights of the process of EasyMKL. It can be an interesting source of information to determine how the algorithm determines which kernels are relatively more important.

From the results it can be observed that k_{MDARC1} usually is the kernel with most weight. This further supports the fact that this kernel function is possibly the best performing one for the problem. However, in the case of MKL with out-of-sample validation and normalization, k_{GA} is the kernel with highest weight and this combination is also the one with higher test accuracy. k_{MDARC0} also receives weights higher than the mean, signaling that it is also important in the construction of the model and adds additional information. Finally, it is worth to comment the impact of the normalization in the weights: in all the cases it decreases the weights of k_{VAR} and k_{MDED}, the worst performing kernel functions.

Method	Kernel weight				
	k_{VAR}	k_{GA}	k_{MDED}	k_{MDARC0}	k_{MDARC1}
MKL OOS	0.135	0.182	0.121	0.134	0.429
MKL OOS Norm.	0.128	0.339	0.110	0.159	0.264
MKL CV	0.125	0.163	0.107	0.158	0.449
MKL CV Norm	0.095	0.246	0.093	0.200	0.366

Table 2. Weights of the different MKL procedures.

The following table contains the experiments performed with exogenous variables and their results. All the experiments are executed using the best performing kernel method for the data, normalized multiple kernel learning using out-of-sample validation technique. Each variable is contained in a data frame that will be transformed into a kernel matrix using the MKL procedure. This creates four data frames (one for each exogenous variable) plus one containing only the endogenous variable (the S&P 500 index) and an additional data frame containing all the variables. In order to extract more information from these variables, four different methods of building the data frames were tested. DF1 only considers the exogenous variables without lags and SPX with four lags is included in each data frame. DF2 adds to the information of DF1 each exogenous variable with a lag of 3; this is motivated by the fact that the resulting data frame will contain more information but without adding too much redundancy. DF3 includes four lags of each exogenous variable. DF4 contains the same information as DF3 for the exogenous variables, but only includes the SPX without lags.

	Accuracy				Weights					
	Train Acc.	Validation Acc.	Test Acc.		SPX	DP	EP	DY	SV	All
DF1	0.767	0.757	0.660	DF1	0.170	0.167	0.167	0.167	0.166	0.162
DF2	0.768	0.757	0.660	DF2	0.171	0.166	0.167	0.166	0.168	0.162
DF3	0.759	0.744	0.680	DF3	0.197	0.140	0.141	0.140	0.194	0.188
DF4	0.744	0.787	0.649	DF4	0.141	0.176	0.181	0.176	0.190	0.136

Table 3. Tests with exogenous variables.

The best performing method is DF3 in terms of test accuracy. DF1 and DF2 are fairly similar, indicating that the inclusion of the third lag does not affect too much the model. DF4 is the worst performing one, which can mean that the model heavily rely on the lags of SPX to predict the output. The weights of DF1 and DF2 are near the mean, with slightly higher weight for the endogenous variable. This result does not mean that the rest of variables are not important to predict the result (that would be represented by weights near zero) but that they are mostly equally important. DF3 shows a shift in the weights, as SV (Stock Variance) is near SPX. The data frame containing all the variables also increases in weight, indicating a better predictive capability of all the variables by including all their lags until 4. Further experimental research shows that the fourth lag of the stock variance provides relevant information that helps the predictive process. Finally, DF4 has its weights shifted towards the exogenous variables. As those data frames lack of lags of SPX it is possible that the information contained in the exogenous variables takes a more active role in the prediction procedure, however the model performs worse in comparison to the rest. It is also worth noting that stock variance is still one of the most relevant variables in this case.

4 Conclusions

In this article, a non-linear approach was applied to the Welch and Goyal experiments [21] to measure the influence of several economic indicators in the prediction of the equity premium. The experiment is designed around several kernel functions for time series that aimed to extract relevant information from these variables and create a predictive model. The experimental procedure was based on selecting the best kernel or combination thereof for this problem, applying the selected model to each of the variables creating several kernel matrices and then obtaining the multiple kernel learning weights of each matrix. These weights indicate the relative importance of each variable.

The results indicate that, in this experimental procedure, the exogenous variables have a relative importance comparable with the S&P 500 index. However, the predictive capabilities of the model are not improved upon the introduction of these values and the weights are not consistent across the different experiments. This instability was previously reported in Welch and Goyal's work and it is now confirmed in our work.

As future works, the influence of each individual variable and the number of lags included could be further explored to find deeper relationships. More and different kernels can also be introduced to enrich the descriptive capabilities of the multiple kernel learning models. In this sense, a finer tuning of the parameters of each kernel should be performed to extract the most of the approach.

References

1. Aiolli, F., Donini, M. (2015). EasyMKL: a scalable multiple kernel learning algorithm. Neuro computing 169, 215–224.
2. Ang, A., and Bekaert, G. (2007). Stock return predictability: Is it there?. Review of Financial studies, 20(3), 651–707.
3. Bach, F. R., Lanckriet, G. R., and Jordan, M. I. (2004). Multiple kernel learning, conic duality, and the SMO algorithm. In Proceedings of the Twenty-First International Conference on Machine learning, p. 6, ACM.
4. Bergmeir, C., Hyndman, R., Koo, B. (2015). A Note on the Validity of Cross-Validation for Evaluating Time Series Prediction. Department of Econometrics and Business Statistics, Working Paper, ISSN 1440-771X.

5. Campbell, John Y., and Shiller, R. J. (1988). The dividend-price ratio and expectations of future dividends and discount factors, Review of Financial Studies 1, 195–228.
6. Campbell, J. Y., and Thompson, S. B. (2008). Predicting excess stock returns out of sample: Can anything beat the historical average?. Review of Financial Studies, 21(4), 1509-1531.
7. Chang, C., Lin, C. (2001) Training ν-Support Vector Classifiers: Theory and Algorithms. Neural Comput. 13 (9), 2119–2147.
8. Cho, Y., Saul, L. (2009). Kernel Methods for Deep Learning. Advances in Neural Information Processing Systems 22, 342–350.
9. Cochrane, John H. (1992). Explaining the variance of price-dividend ratios, Review of Financial Studies 5, 243–280
10. Cochrane, John H. (2006). The dog that did not bark: A defense of return predictability, Review of Financial Studies 21, 1533–1575.
11. Cochrane, John H. (2011). Presidential Address: Discount Rates, The Journal of Finance 56 (4), 1047–1108.
12. Cuturi, M., Vert, J.-P., Birkenes, Ø., Matsui, T. (2007). A kernel for time series based on global alignments. In IEEE Int. Conf. ICASSP 2007, pages II–413. IEEE.
13. Cuturi, M., Doucet, A. (2011). Autoregressive kernels for time series. Technical Report arXiv:1101.0673.
14. Cuturi, M. (2011). Fast global alignment kernels. In Proceedings of the 28th international conference on machine learning (ICML-11), 929–936.
15. Fama, Eugene F., and French, Kenneth R. (1988). Dividend yields and expected stock returns, Journal of Financial Economics 22, 3–25.
16. Hansen, Lars Peter, and Hodrick, Robert J. (1980). Forward exchange rates as optimal predictors of future spot rates: An econometric analysis, Journal of Political Economy 88, 829– 853.
17. Kothari, S. P., and Shanken, J. (1997). Book-to-market, dividend yield, and expected market returns: A time-series analysis. Journal of Financial Economics, 44(2), 169-203.
18. Lettau, M., and Ludvigson, S. (2001). Consumption, aggregate wealth, and expected stock returns. the Journal of Finance, 56(3), 815–849.
19. Peña, M., Arratia, A., and Belanche, L. A. (2016). Multivariate Dynamic Kernels for Financial Time Series Forecasting. In 25th International Conference on Artificial Neural Networks, Springer LNCS 9887, 336–344.
20. Shiller, Robert J. (1981). Do stock prices move too much to be justified by subsequent changes in dividends? American Economic Review 71, 421–436.
21. Welch, I., and Goyal, A. (2008). A comprehensive look at the empirical performance of equity premium prediction. Review of Financial Studies, 21(4), 1455–1508.