ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economics

Bachelor Thesis Econometrics and Operational Research

USING SLOPE-WEIGHTED FORECAST COMBINATIONS TO SOLVE THE FORECAST COMBINATION PUZZLE

Name Student: Finn van der Meer

Student ID number: 504905

Supervisor: Dr. N.W. Koning

Second assessor: Prof. Dr. C. Zhou

Date final version: July 3, 2022

Abstract

The forecast combination puzzle is a phenomenon that the average of a set of forecasters is more robust than a complex combination of those forecasters. In this paper, the SLOPE procedure is used to construct a weighted combination to try and beat the simple average. Kremer et al. (2022) used the SLOPE penalty function in the context of index tracking. They find that SLOPE is able to produce similar results as other state of the art techniques. This was partly because SLOPE is well-equipped to deal with groups in the data, which can also be useful when combining forecasters. Forecasters from the European Central Bank are combined according to the SLOPE procedure and are then compared with the average of these forecasters. The main finding is that, although the SLOPE forecasters perform well, they do not significantly outperform the simple average.

Contents

1 Introduction

The average of a set of forecasters is repeatedly found to outperform more advanced forecasting combinations (Smith & Wallis, 2009). This is unintuitive as the quality of forecasters may differ substantially, and different forecasters of the same variable can be highly correlated. Therefore, this is a widely researched topic. Genre et al. (2013) dive into the shrinking of portfolio weights and Samuels and Sekkel (2017) investigate whether trimming the dataset before averaging improved the forecasts.

Diebold and Shin (2019) use a least absolute shrinkage and selection operator (LASSO) based penalty to resolve this issue. However, this partially-egalitarion LASSO technique is inefficient as it uses two steps. First it gives k selected forecasters a nonzero weight, after which these weights are pulled towards $1/k$. A more elegant solution is the SLOPE penalty function. Because of its design it sets weights to zero and forms groups without having to use two steps.

The sorted $\mathcal{L}_1$ penalized estimator (SLOPE) procedure finds the weights that minimize the difference between a benchmark and a weighted combination of factors. Kremer et al. (2022) use it in the context of Index Tracking (IT), where an index is cloned by making a weighted combination of the constituents of that index. However, it can also be used in a forecasting framework to create a new forecaster that is a weighted combination of other forecasters.

Kremer et al. (2022) found that SLOPE was able to get similar results as other state of the art techniques in the IT framework, which is partly caused by the fact that SLOPE is especially well-equipped to deal with groups in the data. This can prove to be useful when creating a combination of forecasters. When the simple average is taken of a group of forecasters, they are essentially placed in one group with equal weights. Thus, the possibility to create more than one group could prove to be useful in the creation of forecasters. This leads to the main research question of this paper:

"To what extent can the equally weighted forecaster be improved by using a SLOPE-weighted combination of different forecasters?"

The forecasters that are combined are collected by the European Central Bank (ECB) in its Survey of Professional Forecasters (SPF). The weights of these combinations are calculated by the SLOPE procedure. This paper finds that SLOPE-weighted forecasters perform well, yet no significant evidence is found that they outperform the simple average of the ECB forecasters.

The structure of this paper is as follows. First, the existing literature is used to explain why SLOPE is used in a forecast combination framework. Thereafter, the methodology of the SLOPE

penalty is explained. Then, the results found by Kremer et al. (2022) are replicated, after which the SLOPE-weighted forecasts are evaluated, together with a LASSO and equally-weighted portfolio. The conclusion concludes.

2 Literature

2.1 SLOPE procedure

Kremer et al. (2022) use the SLOPE procedure in the context of index tracking and hedge fund replication. Index tracking or hedge fund replication focuses on replicating a benchmark index or a hedge fund with a linear combination of the index constituents or with a set of risk factors.

This has become increasingly popular as more people want to generate similar profits as the benchmarks, but purchasing all of the individual index constituents is time-consuming and hedge funds carry restrictions like not being open to everyone (Giamouridis & Paterlini, 2010). The most common way to construct tracking portfolios is by minimizing a certain tracking error between the clone and the benchmark. For example, Rudolf et al. (1999) take the squared difference between the returns of the benchmarks and its clone.

Creating sparse portfolios through penalty functions

When minimizing this error, the index tracking problem comes down to performing ordinary least squares (OLS). However, because OLS only focuses on minimizing the error, it can occur that the benchmark replica consists of a big number of small fractions of different assets. Because this is costly to implement in practise (Giuzio et al., 2018), sparse portfolio's with a small number of assets are preferred. Penalty functions that penalize the inclusion of new assets into the portfolio are used to create sparse portfolios.

One of the most well known penalty function is the least absolute shrinkage and selection operator (LASSO) penalty, first used by Santosa and Symes (1986) and introduced to statistical literature by Tibshirani (1996). Brodie et al. (2009) found that it is able to produce sparse clones that take transaction costs into account and Giamouridis and Paterlini (2010) show that LASSO can construct sparse and stable hedge fund clones.

However, LASSO still has a number of disadvantages. For example, when assets are equally correlated, LASSO randomly picks one (Bondell & Reich, 2008). The most notable problem with regards to this paper is that LASSO is stuck in the no short-sale area given a budget constraint

(DeMiguel et al., 2009). This means that it is not able to have negative weights for the optimal solutions.

There are non-convex solutions to this but they require a lot of computational power. The most elegant solution is the sorted $\mathcal{L}_1$ penalized estimator (SLOPE) penalty, proposed by Bogdan et al. (2013) and introduced to the index tracking framework by Kremer et al. (2022). Due to the form of the penalty function, SLOPE is able to group strongly correlated assets together. This is a feature that can be very useful in the context of index tracking, which is shown by the fact that SLOPE is able to produce similar results as other state of the art methods (Kremer et al., 2022).

2.2 Forecast combination

As the SLOPE method finds a combination that mimics a certain benchmark, it can also be used to create a forecaster. The benchmark becomes the variable that is being forecasted, and it is mimicked by a linear combination of a number of other forecasters.

There has been plenty of research into the combination of different forecasting techniques. Bates and Granger (1969) were one of the first to show that a combination of forecasts often leads to a better performance. Even a simple average of the forecasters can lead to dramatic performance improvements (Clemen, 1989). More advanced geometric techniques have been used by for example Hsiao and Wan (2014), who used different eigenvector approaches to determine the best forecaster.

SLOPE-weighted forecast combinations

Diebold and Shin (2019) used a LASSO-based approach, the partially-egalitarian (pe)LASSO penalty, to combining forecasts, where they discovered that it did not outperform the average of the forecasts, but they state that new procedures based on the LASSO could prove to be valuable extensions.

A natural upgrade on the LASSO penalty function is the SLOPE penalty function. Contrary to the the peLASSO used by Diebold and Shin (2019), it does not need two steps to set weights to zero and then pull them into groups. This is done at once because of the form of the penalty function.

The grouping procedure of SLOPE was one of the reasons why it performed so well in the index tracking framework Kremer et al. (2022) used it in. However, it could also prove to be a valuable feature in the forecast combination framework. When using the average of different forecasters, all of these forecasters are essentially put into one group with the same weight of 1/(amount of forecasters).

Being able to form more than one group to put these forecasters into could lead to forecasting improvements. Such groups could be formed in a way where bad forecasters are put together in a group with weight zero, while better forecaster are put together in a group with a higher, nonzero, weight. In such a framework, it is still possible to assign the biggest weights to a small number of best performing forecasters.

This all makes me believe that research into using the SLOPE procedure to create a forecaster can be a valuable addition to the preexisting literature.

3 Methodology

The notation used in this section is the same as in Kremer et al. (2022). The goal in both the index tracking and forecasting framework is to mimic a certain benchmarks time series, which is denoted by $T \times 1$ vector $\boldsymbol{Y} = [Y_1, Y_2, \cdots, Y_T]$. This clone is made by making a weighted combination of k different factors. $\boldsymbol{R} = [\boldsymbol{r}^1, \cdots, \boldsymbol{r}^K]$ is the $T \times K$ matrix of those factors, with $\boldsymbol{r}^j = [r_1^j, r_2^j, \cdots, r_T^j]$ the $T \times 1$ the individual time series. For index tracking k different index constituents are considered, for the forecast combinations k different forecasters are combined.

Finding the optimal weights

The optimal weights $\hat{\boldsymbol{w}}$ minimize the squared 2-norm of the difference between the benchmark and the weighted combination, while considering a penalty function and the constraint that the weights are nonnegative and should add up to one:

$$\begin{aligned} \underset{w \in \mathbb{R}^K}{\arg\min} \quad & \|\boldsymbol{Y} - \boldsymbol{R}\boldsymbol{w}\|_2^2 + \rho_\lambda(\boldsymbol{w}), \\ s.t. \quad & \boldsymbol{1'w} = 1, \\ & w_i \geq 0, \forall i = 1, \ldots, K. \end{aligned} \tag{1}$$

In the index tracking framework, $\boldsymbol{1'w} = 1$ is a budget constraint that ensures that the constructed portfolio can be afforded. For forecasting purposes, this constraint is still present because a combination of unbiased forecasters stays unbiased when their weights add up to one. As the forecasts come from the ECB, a certain level of quality can be expected from them and thus unbiasedness of the individual forecasters is a reasonable assumption.

The assumption of non-negative weights also holds for both the index tracking problem and the combining of forecasts. Kremer et al. (2022) use it to ensure that no short sales are performed. In the forecast combination framework, it is also included. This is because, again, a certain level of

quality is expected of the ECB forecasters and thus it is not expected that they would get assigned negative weights. By including this constraint, extra information is added to the model.

Penalty functions

Two different forms of the penalty function $\rho_\lambda(\boldsymbol{w})$ are considered in this paper. Namely, the aforementioned LASSO and SLOPE penalty functions. The LASSO penalty has the following form:

$$\rho_\lambda(\boldsymbol{w}) = \lambda \times \sum_{i=1}^{K} |w_i|, \tag{2}$$

where every weight is punished by the same value of λ. This is not the case for the SLOPE penalty, which is a variation on the LASSO penalty. It looks as follows:

$$\rho_\lambda(\boldsymbol{w}) = \sum_{i=1}^{K} \lambda_i |w|_{(i)} = \lambda_1 |w|_{(1)} + \lambda_2 |w|_{(2)} + \cdots + \lambda_K |w|_{(K)}, \tag{3}$$

$$s.t \qquad \lambda_1 \geq \lambda_2 \geq \cdots \lambda_K \geq 0 \text{ and } |w|_{(1)} \geq |w|_{(2)} \geq \cdots |w|_{(K)},$$

where the lambdas are a sequence of tuning parameters chosen before the minimization process and $|w|_{(i)}$ corresponds to the i-th largest weight of weight vector $\boldsymbol{w}$.

Grouping property of the SLOPE penalty

The SLOPE structure promotes the formation of different groups that have equal weights, which works as follows. Keep in mind that the penalty function $\rho_\lambda(\boldsymbol{w})$ in (3) should be kept small. Increasing the second biggest weight $|w|_{(2)}$ is penalized less than increasing $|w|_{(1)}$ with the same amount as $\lambda_1 > \lambda_2$. Therefore, increasing $|w|_{(2)}$ costs less up until the point where $|w|_{(1)} = |w|_{(2)}$. From there, increasing one of the two weights makes it the largest weight and thus it is punished by the bigger value of λ_1.

Slope promotes grouping because of the high cost to break out of this impasse. It is important to note that SLOPE does not force groups together, when it is not optimal to form a group this will not occur. The choice of lambda also plays a part in this, as bigger gaps between the lambdas means that SLOPE is more likely to form groups. The bigger the gaps between the different lambdas, the higher the cost to break out of the equilibrium, relative to increasing other values, which makes it more likely that groups are formed.

4 Replication of Kremer et al. (2022)

Kremer et al. (2022) use SLOPE to create sparse index clones. After introducing the used methodology, a simulation study is performed to examine the behaviour of SLOPE for generated data. Following this, clones of different hedge funds and S&P indices are created and their performances are evaluated. This section first replicates the simulations. Thereafter, the index clones are replicated and evaluated.

4.1 Simulation study

The simulation study is performed to examine the behaviour of SLOPE on data that has been generated with an underlying group structure. The design of the simulation follows problem (1), where solution $\boldsymbol{w}$ is of interest because it minimizes the difference between a given benchmark vector $\boldsymbol{Y}$ and return matrix $\boldsymbol{R}$.

Simulation design

Kremer et al. (2022) assume that asset return matrix $\boldsymbol{R_{T\times K}}$ follows an underlying risk factor structure, where there are S risk factors that are responsible for the differences between the asset returns. Assume that $\boldsymbol{F_{T\times S}} = [\boldsymbol{f_1}\ \boldsymbol{f_2}\cdots\boldsymbol{f_s}]$ is the risk factor matrix, with f_i the $T\times 1$ vector of returns for the i-th risk factor. With $\boldsymbol{B_{S\times K}}$ as the loading matrix for the individual risk factors, the return matrix $\boldsymbol{R}$ can be constructed as follows, where ϵ is a $T\times K$ matrix of independent normally distributed error terms,

$$\boldsymbol{R} = \boldsymbol{F}\times\boldsymbol{B} + \boldsymbol{\epsilon}. \tag{4}$$

The simulated data has $T = 500$ observations, $K = 99$ amount of assets and $S = 3$ risk factors. To construct matrix $\boldsymbol{R}$, the same procedure as Kremer et al. (2022) is performed. This means that the risk factors f_1, f_2, f_3 are independent and follow the multivariate standard normal $N(0, I_{3\times 3})$ distribution and loadings matrix $\boldsymbol{B_{S\times K}}$ consists of 33 copies of each of the following columns: [0.77 0.64 0]', [0.9 0 -0.42]' and [0 0.31 0.64]'. The columns of the resulting matrix R are normalized, after which it is used to create the benchmark Y:

$$\boldsymbol{Y} = \boldsymbol{Rw} + v, \tag{5}$$

where v is a $T\times 1$ vector of normally distributed error terms, with $v\sim\sigma\times N(0,1)$ and $\sigma = 0.0015$. For $\boldsymbol{w}$, two scenarios are considered. This replication section focuses on the first scenario, where $\boldsymbol{w}$

is chosen in such a way that the assets belonging to group 1, 2 and 3 have coefficients of 0, 2 and 3, respectively. This leads to the following $\boldsymbol{w}$:

$$\boldsymbol{w} = [\underbrace{0\ 0\ \cdots\ 0\ 0}_{Group1}\ \underbrace{2\ 2\ \cdots\ 2\ 2}_{Group2}\ \underbrace{3\ 3\ \cdots\ 3\ 3}_{Group3}].$$

Choosing λ

To use the SLOPE-penalty (3), a non-increasing sequence of tuning parameters is needed. Kremer et al. (2022) follow Bogdan et al. (2013) by choosing the tuning parameters as $\lambda_i = \alpha\Phi^{-1}(1-q_i), \forall i = 1, \ldots, k$, where Φ is the CDF of the standard normal distribution and $q_i = i \times \theta/2k$. The value of θ determines how fast the lambda sequence decreases and is set to 0.1.

What value α should have is up for debate. Bellec et al. (2018) found that α should be slightly larger than σ, while Bogdan and Frommlet (2020) discovered that setting $\alpha < \sigma$ is beneficial for real life data. Based on these results and the value of σ (0.0015), Kremer et al. (2022) vary the value of α on a grid of 12 logarithmically spaced points between $10^{-3.5} \approx 0.0003$ and $10^{-1.7} \approx 0.02$. As a starting point, a sequence with no difference between the lambda values is used, this particular SLOPE penalty is equal to the LASSO penalty. The different lambda sequences can be found in figure 1.

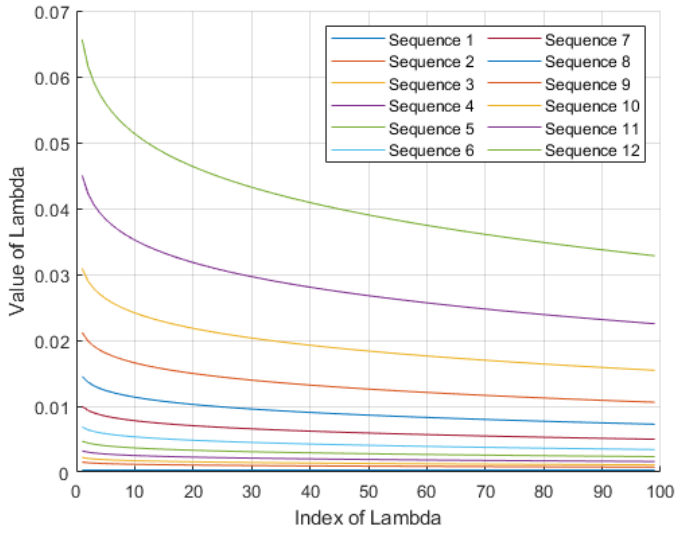


Figure 1: Lambda sequences

Simulation results

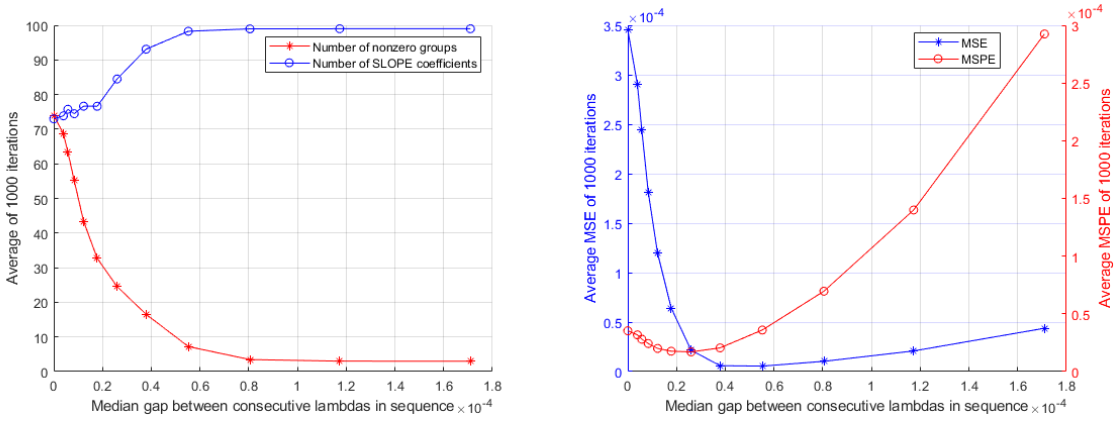
Kremer et al. (2022) perform a simulation where the $\boldsymbol{R}$ and $\boldsymbol{Y}$ matrices are simulated 1000 times for which problem (1) is solved for each of the 12 lambda sequences. This leads to 1000 different $\hat{\boldsymbol{w}}$

vectors for every lambda sequence. The number of nonzero coefficients and the number of groups in the obtained weight vectors are recorded. Figure 2a shows the average of these values over 1000 iterations for every lambda sequence. To measure the performance, the Mean Squared Error (MSE) and the Mean Squared Prediction Error (MSPE) are computed. They are calculated as:

$$MSE(\hat{\boldsymbol{w}}) = \mathbb{E}[||\boldsymbol{w} - \hat{\boldsymbol{w}}||_2^2], \tag{6}$$

$$MSPE(\hat{\boldsymbol{w}}) = \mathbb{E}[||\boldsymbol{R}\boldsymbol{w} - \boldsymbol{R}\hat{\boldsymbol{w}}||_2^2], \tag{7}$$

where $\boldsymbol{w}$ and $\hat{\boldsymbol{w}}$ are respectively the true and estimated portfolio weights and $||\boldsymbol{w}-\hat{\boldsymbol{w}}||_2^2 = \sum_{i=1}^{K}(w_i - \hat{w}_i)^2$. The average over 1000 iterations of both the MSE and MSPE for the different lambda sequences can be found in figure 2b.



(a) Nonzero groups and coefficients

(b) Average MSE & MSPE

Figure 2: Simulation results

Figure 2 and the results derived from it are similar to the results from Kremer et al. (2022). In figure 2a it can be seen that when the gaps between lambdas is zero, and thus there is a LASSO penalty, the number of groups is approximately equal to the number of coefficients. This means that no groups are formed which corresponds with the behaviour that is expected of LASSO. As the gaps become bigger the SLOPE penalty begins to form and assets get grouped together. This is in line with the theoretical findings of section 3. Figure 2b shows that this leads to a better performance as both the MSE and MSPE decrease. Larger gaps lead to a more severe penalty when including new variables, which leads to SLOPE grouping almost all of the assets together. Both the MSE and MSPE values start to increase again when this happens.

4.2 Application on real world data

After applying SLOPE procedure to simulated data, Kremer et al. (2022) use it on real world data where S&P 100, 200 and 500 indices are cloned. This is done by solving problem (1), where the returns of the index in question and the returns of the individual constituents of this index are used as $\boldsymbol{Y}$ and $\boldsymbol{R}$, respectively.

S&P data

The S&P 200 data was difficult to obtain and it focuses on the Australian market, as opposed to the S&P 100 and 500 which focus on the American market. Therefore, this replication section omits the investigation of the S&P 200 index.

The returns of both the S&P 100 and S&P 500 and of their respective constituents were obtained from Datastream for the period of 31 December 2004 until 29 January 2016, which leads to $T = 2890$ return observations. There are stocks that got included into the S&P indices between 2004 and 2016. When this is the case, the data up until the point they got included is missing and thus these stocks are dropped from the dataset. Because of these omissions the S&P 100 has K = 86 assets and the S&P 500 has $K = 409$ assets.

Rolling window explanation

The performance of the produced index clones are investigated by means of a rolling window approach with a window size of $\tau = 750$ daily observations. At time t, the previous 750 observations are used to estimate $\hat{\boldsymbol{w}}_{\boldsymbol{t}}$. Using this vector, the out of sample excess return between the constructed clone and the index can be estimated as: $\boldsymbol{Y}_{\boldsymbol{t+i}} - \boldsymbol{R}_{\boldsymbol{t+i}}\hat{\boldsymbol{w}}_{\boldsymbol{t}}$. This is done for $i = 1, \ldots, n$, after which the rolling window moves forward and drops the first n and adds the next n observations and the process starts again. Choosing $n = 21$ ensures that the portfolio is rebalanced monthly. Because of the amount of observations $T = 2890$ and the values $n = 21$ and $\tau = 750$, the window rolls forwards $M = 101$ times. This leads to 101 different optimal weight vectors $\hat{\boldsymbol{w}}_{\boldsymbol{t}}$.

Choosing λ

Similar as to the simulation study, a lambda sequence is needed for the penalty function and it is constructed in the same way as before. The method of Bogdan et al. (2013) is followed again which leads to tuning parameters $\lambda_i = \alpha\Phi^{-1}(1 - q_i), \forall i = 1, \ldots, k$, where Φ is the CDF of the standard normal distribution and $q_i = i \times \theta/2k$ with $\theta = 0.1$. Kremer et al. (2022) use a grid of 100 linearly spaced alphas, however it is not specified between what values this grid lies. In this paper, the same upper and lower bound of $10^{-3.5} \approx 0.0003$ and $10^{-1.7} \approx 0.02$ from the simulation study are used.

Additionally, a LASSO penalty is also considered, for which the lambdas do not decrease after the initial value is set: $\lambda_1 = \lambda_{LASSO}$.

The optimal tuning parameter is chosen by minimizing a criterion inspired by the Bayesian information criterion (BIC), which focuses on a trade-off between the performance and the amount of active weights. The chosen lambda sequence is the one that minimizes:

$$SC = Mn \times log\left(\frac{\sum_{t=1}^{M}\sum_{i=1}^{n}(\boldsymbol{Y}_{t+1} - \boldsymbol{R}_{t+i}\hat{\boldsymbol{w}}_{\boldsymbol{t}})^2}{Mn}\right) + log(Mn) \times \frac{\sum_{i=1}^{k}\mathbb{I}(w_i \neq 0)}{k}, \tag{8}$$

where $\mathbb{I}(\cdot)$ is an indicator function and thus counts the amount of active weights of the portfolio. The first part keeps the tracking error volatility low, while the second part keeps the amount of active positions low. Equation (8) slightly deviates from the information criterion used by Kremer et al. (2022) as they deviated from the traditional version of the Bayesion Information Criterion (BIC).

Evaluation criteria

With the obtained lambda, optimal weight vector $\hat{\boldsymbol{w}}$ can be calculated. The tracking ability of the clones are then evaluated by looking at the tracking error volatility (TEV) and the tracking error (TE):

$$TEV = \frac{1}{Mn}\sum_{t=1}^{M}\sum_{i=1}^{n}(\boldsymbol{Y}_{t+1} - \boldsymbol{R}_{t+i}\hat{\boldsymbol{w}}_{\boldsymbol{t}})^2, \tag{9}$$

$$TE = \frac{1}{Mn}\sum_{t=1}^{M}\sum_{i=1}^{n}(\boldsymbol{Y}_{t+i} - \boldsymbol{R}_{t+i}\hat{\boldsymbol{w}}_{\boldsymbol{t}}). \tag{10}$$

These can then be used to compute the information ratio, given by the ratio of the TE to the TEV. Because Kremer et al. (2022) are interested in creating a sparse portfolio that is cost efficient, statistics on the total number of active positions (AP) and the average total turnover (TO) are included. These statistics are given by:

$$AP = \frac{1}{M}\sum_{t=1}^{M}\sum_{i=1}^{K}\mathbb{I}(w_{i,t} \neq 0), \tag{11}$$

$$TO = \frac{1}{M}\sum_{t=1}^{M}\sum_{i=1}^{K}|w_{i,t-1} - w_{i,t}|). \tag{12}$$

Because the created portfolios ideally should perform well out of sample, the predictive abilities of the models are analysed as well. The predictive R^2_{OOS} is given by:

$$R^2_{OOS} = 1 - \frac{\sum_{t=1}^{M}\sum_{i=1}^{n}(\boldsymbol{Y}_{t+1} - \boldsymbol{R}_{t+i}\hat{\boldsymbol{w}}_{\boldsymbol{t}})^2}{\sum_{t=1}^{M}\sum_{i=1}^{n}(\boldsymbol{Y}_{t+1})^2}. \tag{13}$$

Empirical results

The values of all the introduced statistics for both the LASSO and SLOPE penalty on the S&P 100 and 500 data can be found in table 1. An optimal strategy has good tracking capabilities with respect to the index. We see that for both the S&P 100 and 500 indices SLOPE has lower values for both the tracking error volatility and the tracking error.

Additionally, a low number of active positions and turnover ratio are desired to create a sparse and cost-efficient clone. LASSO has slightly lower values than SLOPE for these statistics for both the S&P 100 and S&P 500 indices. Finally, a high R^2_{OOS} is desired to evaluate the out of sample performance. Both SLOPE and LASSO show the same value of 0.99 for this criterion for both indices.

	S&P100		**S&P500**	
	LASSO	*SLOPE*	*LASSO*	*SLOPE*
Tracking Error Volatility (in%)	5.10	4.98	3.10	2.90
Tracking Error (in%)	0.90	0.75	1.63	1.48
Information Ratio	0.176	0.151	0.526	0.510
Active Positions	79	81	276	321
Turnover (in%)	0.042	0.045	0.072	0.096
Predicted R^2_{OOS}	0.99	0.99	0.99	0.99

Table 1: Tracking statistics S&P indices

5 Extension: SLOPE-weighted forecasts

Kremer et al. (2022) use the SLOPE procedure for index cloning, where to goal is to clone an index with a combination of its constituents. However, in this section the SLOPE procedure is used to make forecast combinations.

5.1 Data and methods

The data used for this extension is the European Central Bank's (ECB) quarterly survey of professional forecasters (SPF). The SPF is a panel that makes 1 year ahead predictions for the harmonised index of consumer prices (HICP) inflation, real GDP (RGDP) growth and the seasonally adjusted unemployment rates. The survey was launched in the last quarter of 1999 and the most recent

observations are from the first quarter of 2022. The sample is cut off at the beginning of 2020, as to not include the COVID-19 outbreak and its influence on the economic climate because this was an exogenous shock to the market and it is not expected that the forecasts can predict such an event. This means that there are a total of $t = 82$ time periods where the survey is sent to 130 different forecasters.

Cleaning the data

There are many missing responses throughout the survey because forecasters sometimes do not reply. Such an unbalanced dataset cannot be used to come to meaningful conclusions and to solve this a procedure introduced by Genre et al. (2013) is used. First, the forecasters who did not respond in 4 consecutive surveys, which corresponds to one calendar year, are excluded from the data. This leaves 13, 12 and 9 forecasters for HICP, RGDP and unemployment, respectively.

After removing these unresponsive forecasters, there are still several gaps in the data. Genre et al. (2013) introduced the following linear filter to fill these gaps:

$$\hat{y}_{i,t+1} - \bar{y}_{t+1} = \beta_i(\hat{y}_{i,t} - \bar{y}_t) + \epsilon_{i,t+1}, \tag{14}$$

where $\bar{y}_t$ is the average prediction of the forecasters that did respond. Missing observation $\hat{y}_{i,t+1}$ from forecaster i at time $t+1$ can be constructed as $\bar{y}_{t+1} + \beta_i(\hat{y}_{i,t} - \bar{y}_t)$. The β_i values are estimated with an OLS regression and measure to what extend a forecasters predictions are above or below the mean of the other forecasters.

When the first observation of a forecaster is missing, the recursive formula 14 cannot be used and this value will be set to $\bar{y}_1$. In the rare case that the first 2 or even 3 values are missing, these will also be set to $\bar{y}_2$ or $\bar{y}_3$. These values are not used when estimating the β values to prevent a bias towards zero.

Combining the forecasts

With this cleaned and filtered ECB SPF data problem (1) is solved to find the best forecast combination for the three SPF variables. The rolling window approach used to create the forecasts $\hat{y}_{t+1|t}$ are the same as used in Diebold and Shin (2019).

The first 5 observations are used to 'burn in' the forecasts, then for observations 6-20 all the available data is used to create an optimal weight vector $\hat{\boldsymbol{w}}$. For periods $t > 20$, the previous 20 observations are used to create a $\hat{\boldsymbol{w}}$ to construct a forecaster for time $t + 1$.

The SLOPE penalty function (3) is used and the lambda sequence is given by $\lambda_i = \alpha\Phi^{-1}(1 - q_i), \forall i = 1, \ldots, k$. A set of 100 linearly spaced α values is used to create 100 different lambda

sequences. The value of α that leads to the $\hat{\boldsymbol{w}}$'s with the lowest Root Mean Squared Error (RMSE) is used. The RMSE is defined as:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}, \tag{15}$$

where y_i is the true value of the parameter and $\hat{y}_i = R\hat{w}$ its estimator. The RMSE gives the error that a forecaster has when making predictions, and thus low values are preferred.

Mincer-Zarnowitz regression

To evaluate the performance of our different forecasters, Mincer-Zarnowitz regressions, introduced by Mincer and Zarnowitz (1969) are used. A Mincer-Zarnowitz regression is made by regressing the target time series on the forecast as follows:

$$y_{t+1} = \alpha + \beta \hat{y}_{t+1|t} + \epsilon, \tag{16}$$

for which the joint hypothesis of $\alpha = 0$ and $\beta = 1$ is tested. An α of zero means that the forecaster is unbiased, while a β of one means that the forecaster is efficient. These regressions are used to determine if a single forecaster is able to accurately mimic the time series it is forecasting.

Diebold-Mariano test

To compare the performance of different forecasting methods against each other, another method is used. The Diebold-Mariano (DM) test, introduced by Diebold and Mariano (1995), is used to compare the performance of two different forecasting methods against each other. The null hypothesis of the test is that two different forecasters have equal forecast accuracy. This test is conducted by making use of the loss-differential. Let $\hat{y}_1$ and $\hat{y}_2$ be different forecasters for time series y. Then, the residuals of forecaster $i = 1, 2$ at time t are calculated as $e_{i,t+1} = \hat{y}_{i,t+1|t} - y$. With these, the loss differential at time t is defined as:

$$d_t = e_{t,1}^2 - e_{t,2}^2.$$

Under the null hypothesis, the forecasters perform the same and the loss differential has value zero. The DM test tests whether this true against the two-sided alternative hypothesis of the forecasters not performing the same. The DM test statistic is:

$$DM = \frac{\bar{d}}{\sqrt{T^{-1} \times Var(d)}} \sim N(0, 1), \tag{17}$$

where a significant positive (negative) DM test statistic implies that forecaster 1 (2) has a larger error than the other forecaster and thus performs worse.

5.2 Results

In this section, the SLOPE-weighted forecasts are evaluated. They are compared to forecast combinations made with the LASSO penalty and to an equally weighted combination of all the ECB's forecasters.

Figure 3 shows the different SLOPE weighted forecasts and the equally weighted ECB's forecasts against the levels of inflation, real GDP and unemployment. The figure shows that the predictions seem to be somewhat lagged behind the true value of the parameters. This is not unexpected when taking into account that SLOPE chooses combinations of one year ahead forecasts. Additionally, it can be observed that the SLOPE forecaster and the ECB average are similar and that they follow the true value quite closely.

RMSE values

Table 2 shows the RMSE values of the different forecasters for the three forecasted variables. We observe that for the HICP inflation, the SLOPE and LASSO forecasters have a lower RMSE than the ECB average. Between these two, SLOPE has a slightly smaller RMSE than LASSO. Regarding the real GDP, the RMSE values lie much closer together. For the ECB average and LASSO, the values are almost the same, with SLOPE being slightly below them. Finally, for the unemployment forecasts, the ECB average has the lowest RMSE, with SLOPE and LASSO being almost equal.

	HICP	**RGDP**	**UNEMP**
RMSE			
ECB average	0.811	1.427	0.727
LASSO	0.762	1.428	0.740
SLOPE	0.758	1.420	0.739

Table 2: RMSEs for different forecasting methods

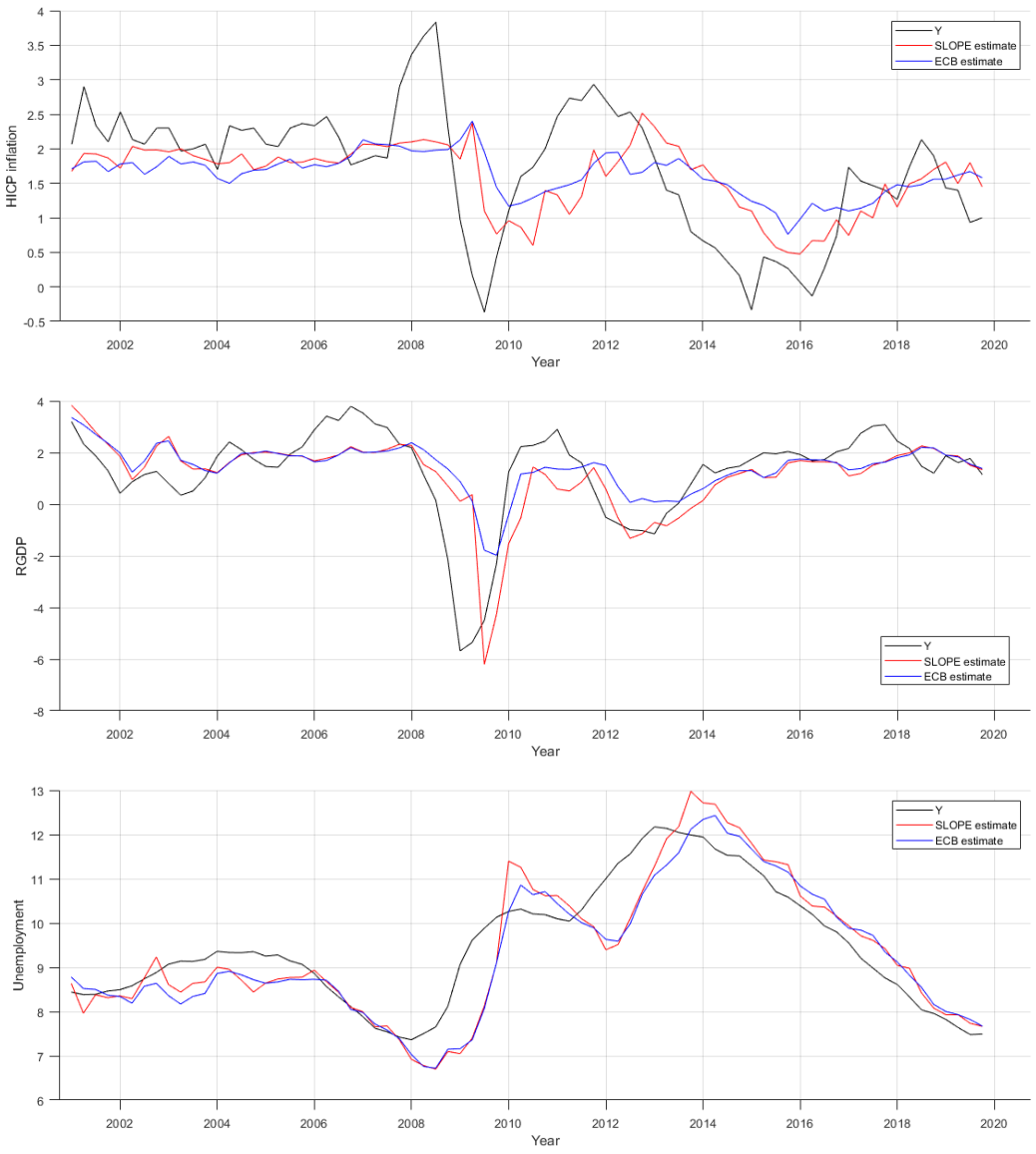


Figure 3: Graphs with the true values and the forecasts of the three different ECB SPF variables

Mincer-Zarnowitz results

The results of the Mincer-Zarnowitz regression can be found in table 3. What stands out is the fact that all three the forecasters predict the HICP inflation well, but are not able to accurately predict the unemployment levels. For SLOPE-weighted HICP forecasts, there is not enough evidence to reject the null-hypothesis of bias and inefficiency at even the 10% level. Likewise, this is also the case for the LASSO forecasts and the ECB average. However, for the ECB average the parameter values lie further away from the null parameters which translates into lower p-values.

For the real GDP forecasters, the SLOPE forecasts still perform well as they do not reject the

null-hypotheses at the 5% level. However, they would reject at the 10% level. This is also the case for the α of the LASSO forecast, but there the null hypothesis of $\beta = 1$ would reject at even the 5% level. Finally, both the α and the β in the Mincer-Zarnowitz regression for the average of the ECB forecasters would not reject at the 5% level and the β does not reject at the 10% level.

The unemployment forecasters do not perform that well according to the Mincer-Zarnowitz regression results. All three the different forecasters reject both the null hypotheses of $\alpha = 0$ and $\beta = 1$, as all the p-values are ≤ 0.001. This seems odd, as the RMSE values for the unemployment forecasts were similar to the RMSEs of the forecasters for the other two variables.

		HICP	**RGDP**	**UNEMP**
SLOPE	α	-0.03 (p = 0.923)	0.37(p = 0.075)	2.27(p < 0.001)
	β	1.09 (p = 0.626)	0.78(p = 0.052)	0.77(p < 0.001)
LASSO	α	-0.01 (p = 0.986)	0.35(p = 0.092)	2.29(p < 0.001)
	β	1.06 (p = 0.720)	0.77(p = 0.043)	0.774(p < 0.001)
ECB average	α	-0.60 (p = 0.240)	-0.59(p = 0.065)	1.82(p < 0.001)
	β	1.42 (p = 0.174)	1.28(p = 0.139)	0.82(p = 0.001)

Table 3: Parameter estimates of the Mincer-Zarnowitz regressions with between brackets the p-values of the α's and β's being equal to 0 and 1, respectively.

Diebold-Mariano results

The Diebold-Mariano statistics for the HICP inflation, real GDP and unemployment forecasts can be found in table 4, 5 and 6, respectively. The lowest p-values can be found in table 4, where the SLOPE and LASSO forecasts have a negative DM test statistic when compared to the ECB average.

However, we observe that none of the tests find enough evidence to reject the null-hypothesis of $d_t = 0$, at any of the usual significance levels. This is somewhat disappointing as it means that there is not enough evidence found to conclude that a SLOPE-weighted combination outperforms the equally-weighted forecaster. Nonetheless, it does mean that the SLOPE weighted forecast combination is not worse than the equally weighted forecast combination, which is known to be a good forecaster.

1 \ 2	LASSO	ECB average
SLOPE	-0.30(p = 0.764)	-1.25(p = 0.212)
LASSO	x	-1.29(p = 0.194)

Table 4: DM statistics of the error of forecast combination (1) minus the error of forecast combination (2) with corresponding p-values for the inflation forecasts.

1 \ 2	LASSO	ECB average
SLOPE	0.47 ($p = 0.642$)	-0.73 (p = 0.465)
LASSO	x	-0.74 (p = 0.458)

Table 5: DM statistics of the error of forecast combination (1) minus the error of forecast combination (2) with corresponding p-values for the inflation forecasts.

1 \ 2	LASSO	ECB average
SLOPE	-1.04(p = 0.296)	0.93(p = 0.351)
LASSO	x	1.10(p = 0.269)

Table 6: DM statistics of the error of forecast combination (1) minus the error of forecast combination (2) with corresponding p-values for the inflation forecasts.

6 Conclusion

The goal of this paper was to answer the research question:

"To what extent can the equally weighted forecaster be improved by using a SLOPE-weighted combination of different forecasters?"

This was done by combining ECB forecasts on inflation, GDP and unemployment according to weights calculated with the SLOPE procedure.

Regarding the inflation and the GDP forecasts, the SLOPE combination has a lower MSPE than the equally-weighted forecaster. However, in forecasting the unemployment, the equally weighted portfolio has a lower RMSE than the SLOPE forecaster.

To formally examine whether one of the two forecasters outperformed the other, a Diebold-Mariano test was performed. However, this test did not find significant proof that one forecaster outperformed the other. Therefore, I cannot say with proof that a SLOPE-weighted forecaster is an improvement over the equally weighted forecaster.

However, for the inflation forecasters, the SLOPE forecasts has a lower RMSE value than the ECB average and the DM test statistic is nearly significant. This, combined with the fact that SLOPE is well-equipped to make groups of forecasters, makes me believe that SLOPE-weighted forecasts could still be a better choice than the simple average of a group of forecasters. To make more use out of the grouping property of SLOPE, a larger set of forecasters could be used to make a SLOPE-weighted forecast. This is something that could be investigated in further research.

References

Bates, J. M., & Granger, C. W. J. (1969). The combination of forecasts. *Journal of the Operational Research Society*, *20*(4), 451–468.

Bellec, P. C., Lecué, G., & Tsybakov, A. B. (2018). SLOPE meets LASSO: Improved oracle bounds and optimality. *The Annals of Statistics*, *46*(6B), 3603–3642.

Bogdan, M., Berg, E. v. d., Su, W., & Candes, E. (2013). Statistical estimation and testing via the sorted l1 norm.

Bogdan, M., & Frommlet, F. (2020). Identifying important predictors in large data bases – multiple testing and model selection.

Bondell, H. D., & Reich, B. J. (2008). Simultaneous regression shrinkage, variable selection, and supervised clustering of predictors with oscar. *Biometrics*, 115–123.

Brodie, J., Daubechies, I., Mol, C. D., Giannone, D., & Loris, I. (2009). Sparse and stable markowitz portfolios. *Proceedings of the National Academy of Sciences*, *106*(30), 12267–12272.

Clemen, R. T. (1989). Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting*, *5*(4), 559–583.

DeMiguel, V., Garlappi, L., Nogales, F. J., & Uppal, R. (2009). A generalized approach to portfolio optimization: Improving performance by constraining portfolio norms. *Management Science*, *55*(5), 798–812.

Diebold, F. X., & Mariano, R. S. (1995). Comparing predictive accuracy. *Journal of Business Economic Statistics*, *13*(3), 253–263.

Diebold, F. X., & Shin, M. (2019). Machine learning for regularized survey forecast combination: Partially-egalitarian lasso and its derivatives. *International Journal of Forecasting*, *35*(4), 1679–1691.

Genre, V., Kenny, G., Meyler, A., & Timmermann, A. (2013). Combining expert forecasts: Can anything beat the simple average? *International Journal of Forecasting*, *29*(1), 108–121.

Giamouridis, D., & Paterlini, S. (2010). Regular(Ized) Hedge Fund Clones. *Journal of Financial Research*, *33*(3), 223–247.

Giuzio, M., Eichhorn-Schott, K., Paterlini, S., & Weber, V. (2018). Tracking hedge funds returns using sparse clones. *Annals of Operations Research*, *266*(1), 349–371.

Hsiao, C., & Wan, S. K. (2014). Is there an optimal forecast combination? [Recent Advances in Time Series Econometrics]. *Journal of Econometrics*, *178*, 294–309.

Kremer, P. J., Brzyski, D., Bogdan, M., & Paterlini, S. (2022). Sparse index clones via the sorted 1-norm. *Quantitative Finance*, *22*(2), 349–366.

Mincer, J. A., & Zarnowitz, V. (1969). The evaluation of economic forecasts. *Economic forecasts and expectations: Analysis of forecasting behavior and performance* (pp. 3–46). NBER.

Rudolf, M., Wolter, H.-J., & Zimmermann, H. (1999). A linear model for tracking error minimization. *Journal of Banking Finance*, *23*(1), 85–103.

Samuels, J. D., & Sekkel, R. M. (2017). Model confidence sets and forecast combination. *International Journal of Forecasting*, *33*(1), 48–60.

Santosa, F., & Symes, W. W. (1986). Linear inversion of band-limited reflection seismograms. *SIAM Journal on Scientific and Statistical Computing*, *7*(4), 1307–1330.

Smith, J., & Wallis, K. F. (2009). A Simple Explanation of the Forecast Combination Puzzle. *Oxford Bulletin of Economics and Statistics*, *71*(3), 331–355.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, *58*(1), 267–288.

Appendix

For both the replication and extension part of this thesis, I made use of the matlab code that Philipp J. Kremer sent which calculates the SLOPE weights.

Replication code

The code for the replication is in map Algo. For the simulation part, run replication_simulation_code. For the S&P data, use replication_spdata_slope_code for the SLOPE results and replication_spdata_lasso_code for the LASSO results. Line 11 in both these programs looks like this:

```
SP_X = xlsread('sp100_2016.xlsx', 'X');
```

where sp100_2016 can be changed to sp500_2016 to get the results for the S&P 500 data. After this, run replication_plotter to create the figures in this paper. This code is not very clean and efficient but it works, the code for the extension is more tidy.

Extension code

First run extension_code.m. The different datasets for the three variables can be called in line 5, which looks as follows:

```
data = xlsread('SPF_data_clean.xlsx','HICP');
```

and to change the variable from inflation, 'HICP' can be changed to 'RGDP' and 'UNEMP'. Note that the unemployment data has zeros in them when a variable is missing, thus

```
X = cleanData(X,max_gap);
```

should be changed towards

```
X = cleanDataUnemp(X,max_gap);
```

When using this code on the unemployment data. After running extension_code, Plotter.m can be used to plot the values of the estimates against the true values and the equal weight portfolio. After this, run Evaluation.m to calculate the evaluation criteria.