Optimization and Engineering (2023) 24:2045–2078
https://doi.org/10.1007/s11081-022-09765-w

RESEARCH ARTICLE

Covariance prediction via convex optimization

Shane Barratt[1] · **Stephen Boyd**[1]

Received: 12 June 2022 / Revised: 17 August 2022 / Accepted: 17 August 2022 /
Published online: 3 September 2022

Abstract
We consider the problem of predicting the covariance of a zero mean Gaussian vector, based on another feature vector. We describe a covariance predictor that has the form of a generalized linear model, *i.e.*, an affine function of the features followed by an inverse link function that maps vectors to symmetric positive definite matrices. The log-likelihood is a concave function of the predictor parameters, so fitting the predictor involves convex optimization. Such predictors can be combined with others, or recursively applied to improve performance.

1 Introduction

1.1 Covariance prediction

We consider data consisting of a pair of vectors, an outcome vector $y \in \mathbf{R}^n$ and a feature vector $x \in \mathcal{X} \subseteq \mathbf{R}^p$, where $\mathcal{X}$ is the feature set. We model y conditioned on x as zero mean Gaussian, *i.e.*, $y \mid x \sim \mathcal{N}(0, \hat{\Sigma}(x))$, where $\hat{\Sigma} : \mathcal{X} \to \mathbf{S}^n_{++}$, the set of symmetric positive definite $n \times n$ matrices. (We will address later the extension to nonzero mean.) Our goal is to fit the covariance predictor $\hat{\Sigma}$ based on observed training data x_i, y_i, $i = 1, \ldots, N$. We judge a predictor $\hat{\Sigma}$ by its average log-likelihood on out of sample or test data,

$$\frac{1}{2\tilde{N}} \sum_{i=1}^{\tilde{N}} \left(-n \log(2\pi) - \log \det \hat{\Sigma}(\tilde{x}_i) - \tilde{y}_i^T \hat{\Sigma}(\tilde{x}_i)^{-1} \tilde{y}_i \right),$$

where $\tilde{x}_i, \tilde{y}_i$, $i = 1, \ldots, \tilde{N}$ is a test data set.

The covariance prediction problem arises in several contexts. As a general example, let y denote the prediction error of some vector quantity, using some prediction

✉ Stephen Boyd
boyd@stanford.edu

[1] Stanford University, 254 Packard EE Building, Stanford, USA

that depends on the feature vector x. In this case we are predicting the (presumed zero mean Gaussian) distribution of the prediction error as a function of the features. In statistics, this is referred to as heteroscedasticity, since the uncertainty in the prediction depends on the feature vector x.

The covariance prediction problem also comes up in vector time series, where i denotes time period. In these applications, the feature vector x_i is known in period i, but the outcome y_i is not, and we are predicting the (presumed zero mean Gaussian) distribution of y_i. In the context of time series, the covariance predictor $\hat{\Sigma}(x_i)$ is also known as the covariance forecast. In time series applications, the feature vector x_i can contain quantities known at time i that are related to y_i, including quantities that are derived directly from past values $y_{i-1}, y_{i-2}, \ldots$, such as the entries of $y_{i-1}y_{i-1}^T$ or some trailing average of them. We will mention some specific examples in Sect. 3.5, where we describe some well known covariance predictors in the time series context.

As a specific example, y_i is the vector of returns of n financial assets over day i, with mean small enough to ignore, or already subtracted. The return vector y_i is not known on day i. The feature vector x_i includes quantities known on day i, such as economic indicators, volatility indices, previous realized trading volumes or returns, and (the entries of) $y_{i-1}y_{i-1}^T$. The predicted covariance $\hat{\Sigma}(x_i)$ can be interpreted as a (time-varying) risk model that depends on the features.

1.2 Parametrizing and fitting covariance predictors

Covariance predictors can range from simple, such as the empirical covariance of the outcome on the training data (which is constant, *i.e.*, does not depend on the features), to very complex, for example a neural network or a decision tree that maps the feature vector to a positive definite matrix. (We will describe many covariance predictors in Sect. 3.) Many predictors include parameters that are chosen by solving an optimization problem, typically maximizing the log-likelihood on the training data, minus some regularization. In most cases this optimization problem is not convex, so we have to resort to heuristic methods that approximately solve it. In a few cases, including our proposed method, the fitting problem is convex, which means it can be reliably globally solved.

In this paper we focus on a predictor that has the same form as a generalized linear model, *i.e.*, an affine function of the features followed by a function interpretable as an inverse link function that maps vectors to symmetric positive definite matrices. The associated fitting problem is convex, and so readily solved globally.

1.3 Outline

In Sect. 2 we observe that covariance prediction is equivalent to finding a feature-dependent linear mapping of the outcome that whitens it, *i.e.*, maps its distribution to $\mathcal{N}(0, I)$. Our proposed covariance predictor is based directly on this observation. The interpretation also suggests that multiple covariance prediction methods can be iterated, which we will see in examples leads to improved performance. In

Sect. 3 we outline the large body of related previous work. We present our proposed method, the regression whitener, in Sect. 4, and give some variations and extensions of the method in Sect. 5. In Sects. 6 and 7 we illustrate the ideas and our method on two examples, a financial time series and a machine learning residual problem.

2 Feature-dependent whitening

In this section we show that the covariance prediction problem is equivalent to finding a feature-dependent linear transform of the data that whitens it, *i.e.*, results (approximately) in an $\mathcal{N}(0, I)$ distribution.

Given a covariance predictor $\hat{\Sigma}$, define $L : \mathcal{X} \to \mathcal{L}$ as

$$L(x) = \mathbf{chol}(\hat{\Sigma}(x)^{-1}) \in \mathcal{L},$$

where **chol** denotes the Cholesky factorization, and $\mathcal{L}$ is the set of $n \times n$ lower triangular matrices with positive diagonal entries. For $A \in \mathbf{S}^n_{++}$, $L = \mathbf{chol}(A)$ is the unique $L \in \mathcal{L}$ that satisfies $LL^T = A$. Indeed, $\mathbf{chol} : \mathbf{S}^n_{++} \to \mathcal{L}$ is a bijection, with inverse mapping $L \mapsto LL^T$. We can think of L as a feature-dependent linear whitening transformation for the outcome y, *i.e.*, $z = L(x)^T y$ should have an $\mathcal{N}(0, I)$ distribution.

Conversely we can associate with any feature-dependent whitener $L : \mathcal{X} \to \mathcal{L}$ the covariance predictor

$$\hat{\Sigma}(x) = (L(x)L(x)^T)^{-1} = L(x)^{-T}L(x)^{-1}.$$

The feature-dependent whitener is just another parametrization of a covariance predictor.

2.1 Interpretation of Cholesky factors

Suppose $y \sim \mathcal{N}(0, (LL^T)^{-1})$, with $L \in \mathcal{L}$. The coefficients of L are closely connected to the prediction of y_i (here meaning the ith component of y) from $y_{i+1}, \ldots, y_n$. Suppose the coefficients $a_{i,i+1}, \ldots, a_{i,n}$ minimize the mean-square error

$$\mathbf{E}\left(y_i - \sum_{j=i+1}^{n} a_{i,j}y_j\right)^2.$$

Let J_i denote the minimum value, *i.e.*, the minimum mean-square error (MMSE) in predicting y_i from $y_{i+1}, \ldots, y_n$. We can express the entries of L in terms of the coefficients $a_{i,j}$ and J_i.

We have $L_{ii} = 1/\sqrt{J_i}$, *i.e.*, L_{ii} is the inverse of the standard deviation of the prediction error. The lower triangular entries of L are given by

$$L_{ji} = -L_{ii}a_{i,j}, \quad j = i + 1, \ldots, n.$$

This interpretation has been noted in several places, *e.g.*, (Pourahmadi 2011). An interesting point was made in Wei and Pourahmadi (2003), namely that an approach like this "reduces the difficult and non-intuitive task of modelling a covariance matrix to the more familiar task of modelling $n-1$ regression problems".

2.2 Log-likelihood

For future reference, we note that the log-likelihood of the sample x, y with whitener $L(x)$, and associated covariance predictor $\hat{\Sigma}(x) = L(x)^{-T}L(x)^{-1}$, is

$$\begin{aligned} &-(n/2)\log(2\pi) - (1/2)\log\det\hat{\Sigma}(x) - (1/2)y^T\hat{\Sigma}(x)^{-1}y \\ &\quad = -(n/2)\log(2\pi) + \sum_{j=1}^{n}\log L(x)_{jj} - (1/2)\|L(x)^T y\|_2^2. \end{aligned} \tag{1}$$

The log-likelihood function (1) is a concave function of $L(x)$ (Boyd and Vandenberghe 2004).

2.3 Iterated covariance predictor

The interpretation of a covariance predictor as a feature-dependent whitener leads directly to the idea of iterated whitening. We first find a whitener L_1, to obtain the (approximately) whitened data $z_i^{(1)} = L_1(x_i)^T y_i$. We then find a whitener L_2 for the data $z_i^{(1)}$ to obtain $z_i^{(2)} = L_2(x_i)^T z_i^{(1)}$. We continue this way K iterations to obtain our final whitened data

$$z_i^{(K)} = L_K(x_i)^T \cdots L_1(x_i)^T y_i.$$

This composition of K whiteners is the same as the whitener

$$L(x) = L_1(x)\cdots L_K(x) \in \mathcal{L},$$

with associated covariance predictor

$$\hat{\Sigma}(x) = L_1(x)^{-T}\cdots L_K(x)^{-T}L_K(x)^{-1}\cdots L_1(x)^{-1}.$$

The log-likelihood of sample x, y for this iterated predictor is

$$-(n/2)\log(2\pi) + \sum_{k=1}^{K}\sum_{j=1}^{n}\log(L_k(x))_{jj} - (1/2)\|L_K(x)^T\cdots L_1(x)^T y\|_2^2.$$

This function is multi-concave, *i.e.*, concave in each $L_k(x)$, with the others held constant, but not jointly in all of them.

Our examples will show that iterated whitening can improve the performance of covariance prediction over that of the individual whiteners. Iterated whitening is

related to the concept of boosting in machine learning, where a number of weak learners are applied in succession to create a strong learner (Freund and Schapire 1996).

3 Previous work

In this section we review the very large body of previous and related work, which goes back many years, using the notation of this paper when possible.

3.1 Heteroscedasticity

The ordinary linear regression method assumes the residuals have constant variance. When this assumption is violated, the data or model is said to be heteroscedastic, meaning that the variance of the errors depends on the features. There exist a number of tests to check whether this exists in a dataset (Anscombe 1961; Cook and Weisberg 1983). A common remedy for heteroscedasticity once it is identified is to apply an invertible function to the outcome (when it is positive), *e.g.*, $\log(y)$, $1/y$, or $\sqrt{y}$, to make the variances more constant or homoscedastic (Davidian and Carroll 1987). In general, one needs to fit a separate model for the (co)variance of the prediction residuals, which can be done in a heuristic way by doing a linear regression on the absolute or squared value of the residuals (Davidian and Carroll 1987).

3.2 General examples

Here we describe some of the covariance predictors that have been proposed. We focus on how the predictors are parametrized and fit to training data, noting when the fitting problem is convex. In some cases we report on methods originally developed for fitting a constant covariance, but are readily adapted to fitting a covariance predictor.

3.3 Constant predictor

The simplest covariance predictor is constant, *i.e.*, $\hat{\Sigma}(x) = \Sigma$ for some $\Sigma \in \mathbf{S}^n_{++}$. The simplest way to choose Σ given training data is the empirical covariance, which maximizes the log-likelihood of the training data. More sophisticated estimators of a constant covariance employ various types of regularization (Friedman et al. 2008), or impose special structure on Σ, such as being diagonal plus low rank (Rubin and Thayer 1982) or having a sparse inverse (Dempster 1972; Friedman et al. 2008). Many of these predictors are fit by solving a convex optimization problem, with variable $\hat{\Sigma}(x)^{-1}$, the precision matrix.

3.4 Diagonal predictor

Another simple predictor has diagonal covariance, of the form

$$\hat{\Sigma}(x) = \mathbf{diag}(\exp(Ax + b)), \tag{2}$$

where A and b are the predictor parameters, and exp is elementwise. With this predictor the log-likelihood is a concave function of A and b. (In fact, the log-likelihood is separable across the rows of A and b.) Fitting a diagonal predictor by maximizing log-likelihood, minus a convex regularizer, is a convex optimization problem. This is a special case of Pourahmadi's LDL^T approach where $L = I$ (Pourahmadi 1999). This covariance predictor for the special case of $n = 1$ was implemented in the R package `lmvar` (Posthuma Partners 2019).

This simple diagonal predictor can be used in an iterated covariance predictor, preceded by a constant whitener. For example, we start with a constant base covariance Σ^{const}, with eigenvalue decomposition $\Sigma^{\text{const}} = U\mathbf{diag}(\lambda)U^T$. The iterated predictor has the form

$$\hat{\Sigma}(x) = U\mathbf{diag}(\lambda \circ \exp(Ax + b))U^T,$$

where $\circ$ denotes the elemementwise (Hadamard) product, and A and b are our predictor parameters. In this covariance prediction model we fix the eigenvectors of the (base, constant) covariance, and scale the eigenvalues based on the features. Fitting such a predictor is a convex optimization problem.

3.4.1 Cholesky and LDL^T predictors

Several authors use the Cholesky parametrization of positive definite matrices or the closely related LDL^T factorization of Σ or Σ^{-1}. Perhaps the first to do this was Williams, who in 1996 proposed making the output of a neural network the lower triangular entries and logarithm of the diagonals of the Cholesky decomposition of the inverse covariance matrix (Williams 1996). He used the log-likelihood as the objective to be maximized, and provided the partial derivatives of the objective with respect to the network outputs. The associated fitting problem is not convex. Williams's original work was repeated without citation in Dorta et al. (2018). William's original work was expanded upon and interpreted by Pourahmadi in a series of papers (Pourahmadi 1999, 2000); a good summary of these papers can be found in Pourahmadi (2011). A number of regularization functions for this problem have been considered in Huang et al. (2006). However, to the best of the authors' knowledge, none of these problems is convex, but some are bi-convex, for example Pourahmadi's LDL^T formulation, which has a log-likelihood that is concave in L, and also in the variables $\log D_{ii}$, but not in both sets of variables. Some of the aforementioned methods have been implemented in the R package `jmcm` (Jianxin 2021).

3.4.2 Linear covariance predictors

Some methods proposed by researchers to fit a constant covariance can be readily extended to fit a covariance predictor, *i.e.*, one that depends on the feature vector x. For example, the linear covariance model (Anderson 1973) fits a constant covariance

$$\hat{\Sigma} = \sum_{k=1}^{K} \alpha_k \Sigma_k, \tag{3}$$

where $\Sigma_1, \ldots, \Sigma_K$ are known symmetric matrices and $\alpha_1, \ldots, \alpha_K \in \mathbf{R}$ are coefficients that are fit to data. Of course, $\alpha_1, \ldots, \alpha_K$ must be chosen such that Σ is positive definite; a sufficient condition, when $\Sigma_k \in \mathbf{S}^n_{++}$, is $\alpha \geq 0$ (elementwise), $\alpha \neq 0$. This form is readily extended to be a covariance predictor by making the coefficients α_i functions of x, for example affine, $\alpha = Ax + b$, where A and b are model parameters. (We ignore here the constraint on α, discussed in Sect. 4.1.) For this linear parametrization of the covariance matrix, the log-likelihood is not a concave function of α, so fitting such a predictor is not a convex optimization problem.

Using the inverse covariance or precision matrix, the natural parameter in the exponential family representation of a Gaussian distribution, we do obtain a concave log-likelihood function. The model for a constant covariance is

$$\hat{\Sigma} = \left(\sum_{k=1}^{K} \alpha_k \theta_k \right)^{-1}$$

where $\theta_1, \ldots, \theta_K \in \mathbf{S}^n_{++}$, and α_k are the parameters to be fit, with the constraint $\alpha \geq 0$, $\alpha \neq 0$. The log-likelihood for a sample x, y is

$$-n \log(2\pi) + \log \det \left(\sum_{k=1}^{K} \alpha_k \theta_k \right) - y^T \left(\sum_{k=1}^{K} \alpha_k \theta_k \right) y,$$

which is a concave function of α. This model is readily extended to give a covariance predictor using $\alpha = Ax + b$, where A and b are model parameters. (Here too we must address the issue of the constraints on α.) Fitting the parameters A and b is a convex optimization problem.

Several methods can be used to find a suitable basis $\Sigma_1, \ldots, \Sigma_K$ or $\theta_1, \ldots, \theta_K$. For example, we can run a k-means like algorithm that alternates between assigning data points to the covariance or precision matrix that has highest likelihood, and updating each matrix by maximizing likelihood (possibly minus a regularizer) using the data assigned to it.

3.4.3 Log-linear covariance predictors

Another example of a constant covariance method that can be readily extended to a covariance predictor is the log-linear covariance model. The 1992 paper by Leonard and Hsu Chiu et al. (1996) propose using the matrix exponential, which maps the

vector space $\mathbf{S}^n$ ($n \times n$ symmetric matrices) onto $\mathbf{S}^n_{++}$, for the purpose of fitting a constant covariance matrix. To extend this to covariance prediction, we take

$$\hat{\Sigma}(x) = \exp Z(x), \qquad Z(x) = Z_0 + \sum_{i=1}^{m} x_i Z_i,$$

where $Z_0, \ldots, Z_m$ are (symmetric matrix) model parameters.

The log-likelihood for a sample x, y is

$$-n \log(2\pi) - \mathbf{Tr} Z(x) - y^T (\exp Z(x))^{-1} y,$$

which unfortunately is not concave in the parameters. In a 1999 paper, Williams proposed using the matrix exponential as the final layer in a neural network that predicts covariances (Williams 1999), *i.e.*, the neural network maps x to (the symmetric matrix) $Z(x)$.

3.4.4 Hard regimes and modes

Covariance predictors can be built from a finite number of given covariance matrices, Σ_k, $k = 1, \ldots, K$. The index k is often referred to as a (latent, unobserved) mode or regime. The predictor has the form

$$\hat{\Sigma}(x) = \Sigma_k, \qquad k = \phi(x),$$

where $\phi : \mathcal{X} \to \{1, \ldots, K\}$ is a K-way classifier, tuned with some parameters. We do not know a parametrization of classifiers for which the log-likelihood is concave, but there are several heuristics that can be used to fit such a model. This regime model is a special case of a linear covariance predictor described above, when the coefficients α are restricted to be unit vectors.

One method proceeds as follows. Given the matrices $\Sigma_1, \ldots, \Sigma_K$, we assign to each data sample x, y the value of k that maximizes the likelihood, *i.e.*, the regime that best explains it. We then fit the classifier ϕ to the data pairs x, k. When the classifier is a tree, we obtain a covariance tree, with each leaf associated with one of the regime covariances.[1]

To also fit the regime covariance matrices $\Sigma_1, \ldots, \Sigma_K$, we fix the classifier, and then fit each Σ_k to the data points with $\phi(x) = k$. This procedure can be iterated, analogous to the k-means algorithm.

3.4.5 Soft regimes or modes

We replace the hard classifier described above with a soft classifier

$$\phi : \mathcal{X} \to \{\pi \in \mathbf{R}^K \mid \pi \geq 0,\ \mathbf{1}^T \pi = 1\}.$$

[1] Robert Tibshirani, personal communication.

We can interpret π_k as the probability of regime k, given x. We form our prediction as a mixture of the (given) regime precision matrices,

$$\hat{\Sigma}(x) = \left(\sum_{k=1}^{K} \phi(x)_k \Sigma_k^{-1} \right)^{-1},$$

(which has the same form as a linear covariance predictor with precision matrices, described above). With this predictor, the log-likelihood is a concave function of $\phi(x)$, so when ϕ is an affine function of x, *i.e.*, $\phi(x) = Ax + b$, the fitting problem is convex. (Here we have $\mathbf{1}^T b = 1$ and $\mathbf{1}^T A = 0$, which implies that $\mathbf{1}^T \phi(x) = 1$ for all x, and we ignore the issue that we must have $Ax + b \geq 0$, which we address in Sect. 4.1.)

A more natural soft predictor is multinomial logistic regression, with

$$\phi(x) = \frac{\exp q}{\mathbf{1}^T \exp q}, \qquad q = Ax + b,$$

where A and b are parameters. With this parametrization, the log-likelihood is not concave in A and b, so fitting such a predictor is not a convex optimization problem.

3.4.6 Laplacian regularized stratified covariance predictor

Laplacian regularized stratified models, described in Tuck et al. (2021a), Tuck et al. (2021b), Tuck and Boyd (2020), can be used to develop a covariance predictor. To do this, one bins x into K categories, and gives a possibly different covariance matrix for each of the K bins. (This is the same as a hard regime model with the binning serving as a very simple classifier that maps x to $\{1, \ldots, K\}$.) The predictor is parametrized by the covariance matrices $\Sigma_1, \ldots, \Sigma_K$; the log-likelihood is concave in the precision matrices $\Sigma_1^{-1}, \ldots, \Sigma_K^{-1}$. From the log-likelihood we subtract a Laplacian regularizer that encourages the precision matrices associated with neighboring bins to be close. Fitting such a predictor is a convex optimization problem.

3.4.7 Local covariance predictors

We mention one more natural covariance predictor, based on the idea of a local model (Cleveland and Devlin 1988). We describe here a simple version. The predictor uses the full set of training data, x_i, y_i, $i = 1, \ldots, N$. The covariance predictor is

$$\hat{\Sigma}(x) = \sum_{i=1}^{N} \alpha_i y_i y_i^T, \qquad \alpha_i = \frac{\phi(\|x - x_i\|_2)}{\sum_{j=1}^{N} \phi(\|x - x_j\|_2)},$$

where $\phi : \mathbf{R}_+ \to \mathbf{R}_{++}$ is a radial kernel function. The most common choice is the Gaussian kernel, $\phi(u) = \exp(-u^2/\sigma^2)$, where σ is a characteristic distance parameter. (One variation is to take $\alpha_i = 1/K$ for the K-nearest neighbors of x among $x_1, \ldots, x_N$, and zero otherwise, with K *n*.) We recognize this as a special case of a

linear covariance predictor (3), with a specific choice of the mapping from x to the coefficients, and $\Sigma_i = y_y y_i^T$.

3.5 Time series covariance forecasters

Here we assume that i denotes time period or epoch. At time i, we know the previous realized values $y_{i-1}, y_{i-2}, \ldots$, so functions of them can appear in the feature vector x_i. We write $\hat{\Sigma}(x_i)$ as $\hat{\Sigma}_i$.

3.5.1 SMA

Perhaps the simplest covariance predictor for a time series (apart from the constant predictor) is the simple moving average (SMA) predictor, which averages M previous values of $y_i y_i^T$ to form $\hat{\Sigma}_i$,

$$\hat{\Sigma}_i = \frac{1}{M} \sum_{j=1}^{M} y_{i-j} y_{i-j}^T.$$

Here M is called the memory of the predictor. The SMA predictor follows the recursion

$$\hat{\Sigma}_{i+1} = \hat{\Sigma}_i + \frac{1}{M}(y_i y_i^T - y_{i-M} y_{i-M}^T).$$

Fitting an SMA predictor does not explicitly involve solving a convex optimization problem, but it does maximize the (concave) log-likelihood of the observations y_{i-j}, $j = 1, \ldots, M$.

3.5.2 EWMA

The exponentially weighted moving average (EWMA) predictor uses exponentially weighted previous values of $y_i y_i^T$ to form $\hat{\Sigma}_i$,

$$\hat{\Sigma}_i = \alpha_i \sum_{j=1}^{i-1} \gamma^{i-j} y_j y_j^T, \qquad \alpha_i = \left(\sum_{j=1}^{i-1} \gamma^j \right)^{-1}, \tag{4}$$

where $\gamma \in (0, 1]$ is the forgetting factor, often specified by the half-life $T^{\text{half}} = -(\log 2)/(\log \gamma)$ (Hawkins and Maboudou-Tchao 2008; Harper 2009). This predictor follows the recursion

$$\hat{\Sigma}_{i+1} = \gamma \frac{\alpha_{i+1}}{\alpha_i} \hat{\Sigma}_i + \alpha_{i+1} y_i y_i^T.$$

The EWMA covariance predictor is widely used in finance (Longerstaey and Spencer 1996; Menchero et al. 2011). Like SMA, the EWMA predictor maximizes a (concave) weighted likelihood of past observations.

3.5.3 ARCH

The autoregressive conditional heteroscedastic (ARCH) predictor (Engle 1982) is a variance predictor (*i.e.*, $n = 1$) that uses features $x_i = (y_{i-1}^2, \ldots, y_{i-M}^2)$ and has the form

$$\hat{\Sigma}_i = \alpha_0 + \sum_{j=1}^{M} \alpha_j y_{i-j}^2,$$

where $\alpha_j \geq 0$, $j = 0, \ldots, M$, and M is the memory or order of the predictor. In the original paper on ARCH, Engle also suggested that external regressors could be used as well to predict the variance, which is readily included in the predictor above. A one-dimensional SMA model is a special case of an ARCH model with $\alpha_0 = 0$ and $\alpha_j = 1/M$. The log-likelihood is not a concave function of the parameters $\alpha_0, \ldots, \alpha_M$, so fitting an ARCH predictor requires solving a nonconvex optimization problem.

3.5.4 GARCH

The generalized ARCH (GARCH) (Bollerslev 1986) model, originally introduced by Bollerslev, is a generalization of ARCH that includes prior predicted values of the variance in the features. It has the form

$$\hat{\Sigma}_i = \alpha_0 + \sum_{j=1}^{M} \alpha_j y_{i-j}^2 + \sum_{j=1}^{M} \beta_j \hat{\Sigma}_{i-j},$$

where α_i and β_i are nonnegative parameters. The SMA and EWMA models with $n = 1$ are both special cases of a GARCH model. Like ARCH, the log-likelihood function for the GARCH model is not concave, so fitting it requires solving a nonconvex optimization problem.

3.5.5 Multivariate GARCH

While the original GARCH model is for one-dimensional y_i, it has been extended to multivariate time series. For example, the diagonal GARCH model (Bollerslev et al. 1988) uses a separate GARCH model for each entry of y_i, the constant correlation GARCH model (Bollerslev 1990) assumes a constant correlation between the entries of y_i, and the BEKK model (named after Babba, Engle, Kraft, and Kroner) (Engle and Kroner 1995) is a generalization of all of the above models. There exist many other GARCH variants (see, *e.g.*, (Francq and Zakoian 2019) and the references therein). None of these predictors have a concave log-likelihood, so fitting them involves solving a nonconvex optimization problem.

3.5.6 Time-varying factor models

In a covariance factor model, we regress y_i on some factors w_i, perhaps with exponential weighting, to get $y_i = F_i w_i + \epsilon_i$ (Sect. 3, Grinold and Kahn 2000). Assuming $w_i \sim \mathcal{N}(0, \Sigma^{\text{fact}})$ and $\epsilon_i \sim \mathcal{N}(0, \mathbf{diag}(d_i))$, leads us to the time-varying factor covariance model (Liangjun and Wang 2017)

$$\hat{\Sigma}_i = F_i \Sigma^{\text{fact}} F_i^T + \mathbf{diag}(d_i).$$

The methods of this paper can be used to form a covariance predictor for the factors, which can also depend on features.

3.5.7 Hidden Markov regime models

Hard regime models can be used in the context of time series, with a Markov model for the transitions among regimes (Bilmes 1998). One form for this predictor estimates the probability distribution of the current latent state or regime, and then forms a weighted sum of the precision matrices as our estimate.

4 Regression whitener

In this section we describe a simple feature-dependent whitener, in which $L(x)$ is an affine function of x,

$$\mathbf{diag}(L(x)) = Ax + b, \qquad \mathbf{offdiag}(L(x)) = Cx + d,$$

where **diag** gives the vector of diagonal entries, and **offdiag** gives the strictly lower triangular entries in some fixed order. The regression model coefficients are

$$A \in \mathbf{R}^{n \times p}, \quad b \in \mathbf{R}^n, \quad C \in \mathbf{R}^{k \times p}, \quad d \in \mathbf{R}^k,$$

with $k = n^2/2 - n/2$ denoting the number of strictly lower triangular entries of an $n \times n$ matrix. The total number of parameters in our model is

$$np + n + kp + k = \frac{n(n+1)}{2}(p+1). \tag{5}$$

Our model parameters can be assembled into a single $\frac{n(n+1)}{2} \times (p+1)$ parameter matrix

$$P = \begin{bmatrix} A & b \\ C & d \end{bmatrix}.$$

The top n rows of P give the diagonal of L; its last column gives the constant or offset part of the model, *i.e.*, $L(0)$.

The log-likelihood of the regression whitener is a concave function of the parameters (A, b, C, d). We have already noted that the log-likelihood (1) is a

concave function of $L(x)$, which in turn is a linear function of the parameters (A, b, C, d). The composition of a concave function and a linear function is concave and the sum (over the training samples) preserves concavity (Sect. 3.2, Boyd and Vandenberghe 2004).

With the regression whitener, the precision matrix $\hat{\Sigma}(x)^{-1} = L(x)L(x)^T$ is a quadratic function of the feature vector x; its inverse, the covariance $\hat{\Sigma}(x)$, is a more complex function of x.

4.1 The issue of positive diagonal entries

To have $L(x) \in \mathcal{L}$ for all $x \in \mathcal{X}$, we need $Ax + b > 0$ (elementwise) for all $x \in \mathcal{X}$. When $\mathcal{X} = \mathbf{R}^p$, this holds only when $A = 0$ and $b > 0$. Such a whitener, which has fixed diagonal entries but lower triangular entries that can depend on x, can still have value, but this is a strong restriction. The condition that $Ax + b > 0$ for all $x \in \mathcal{X}$ is convex in (A, b), and leads to a tractable constraint on these coefficients for many choices of the feature set $\mathcal{X}$.

4.1.1 Box features

Perhaps the simplest case is $\mathcal{X} = \{x \mid \|x\|_\infty \leq 1\}$, *i.e.*, the unit box. This means that *all features lie between* -1 *and* 1. This can be ensured in several reasonable ways. First, we can simply clip or Winsorize our raw features $\tilde{x}$, using

$$x = \mathbf{clip}(\tilde{x}) = \min\{1, \max\{-1, \tilde{x}\}\}$$

(interpreted elementwise). Another reasonable approach is to map the values of $\tilde{x}_i$ (the ith component of x) into $[-1, 1]$, for example, by taking $x_i = (2)\mathbf{quantile}_i(\tilde{x}_i) - 1$, where $\mathbf{quantile}_i(\tilde{x}_i)$ is the quantile of $\tilde{x}_i$. Another approach is to scale the values of $\tilde{x}_i$ by its minimum and maximum by taking

$$x_i = 2\frac{\tilde{x}_i - m_i}{M_i - m_i} - 1,$$

where m_i and M_i are the smallest and largest values (elementwise) of x_i in the training data. (When this is used on data not in the training set we would also clip the result of the scaling above to $[-1, 1]$.) We will assume from now on that $\mathcal{X} = \{x \mid \|x\|_\infty \leq 1\}$.

As a practical matter we work with the non-strict inequality $Ax + b \geq \epsilon$ for all $x \in \mathcal{X}$, where $\epsilon > 0$ is given, and the inequality is meant elementwise. The requirement that $Ax + b \geq \epsilon$ for all $\|x\|_\infty \leq 1$ is equivalent to

$$\|A\|_{\text{row},1} \leq b - \epsilon, \tag{6}$$

where $\|A\|_{\text{row},1} \in \mathbf{R}^n_+$ is the vector of ℓ_1 norms of the rows of A, *i.e.*,

$$\left(\|A\|_{\text{row},1}\right)_i = \sum_{j=1}^{p} |A_{ij}|.$$

The constraint (6) is a convex (polyhedral) constraint on (A, b).

4.2 Fitting

Consider a training data set $x_1, \ldots, x_N$, $y_1, \ldots, y_N$. We will choose (A, b, C, d) to maximize the log-likelihood of the training data, minus a convex regularizer $R(A, b, C, d)$, subject to the constraint (6).

This leads to the convex optimization problem

$$\begin{array}{ll} \text{maximize} & (1/N)\sum_{i=1}^{N}\left(\sum_{j=1}^{n}\log(L_i)_{jj} - (1/2)\|L_i^T y_i\|_2^2\right) - R(A,b,C,d) \\ \text{subject to} & \mathbf{diag}(L_i) = Ax_i + b, \quad i = 1,\ldots,N, \\ & \mathbf{offdiag}(L_i) = Cx_i + d, \quad i = 1,\ldots,N, \\ & \|A\|_{\text{row},1} \leq b - \epsilon, \end{array} \tag{7}$$

with variables A, b, C, d. We note that the first term in the objective guarantees that $L(x)_{jj} > 0$ for all training feature values $x = x_i$; the last and stronger constraint ensures that $L(x)_{jj} \geq \epsilon$ for *any* feature vector in $\mathcal{X}$, *i.e.*, $\|x\|_\infty \leq 1$.

4.3 Regularizers

There are many useful regularizers for the covariance prediction problem (7), a few of which we mention here.

4.3.1 Trace inverse regularization

Several standard regularizers used in covariance fitting can be included in R. For example trace regularization of the precision matrix, on the training data, is given by

$$\lambda \frac{1}{N}\sum_{i=1}^{N} \mathbf{Tr}\hat{\Sigma}(x_i)^{-1},$$

where $\lambda > 0$ is a hyper-parameter. This can be expressed in terms of our coefficients as

$$R(A,b,C,d) = \lambda \frac{1}{T}\sum_{t=1}^{T} \|L_i\|_F^2,$$

i.e., ℓ_2-squared regularization on L_i. This can be expressed directly in terms of A, b, C, D as

$$\lambda \frac{1}{T} \sum_{t=1}^{T} \left(\|Ax_i + b\|_2^2 + \|Cx_i + d\|_2^2 \right).$$

We can simplify this regularizer, and remove its dependence on the training data, by assuming that the entries of the features are approximately independent and uniformly distributed on $[-1, 1]$. This leads to the approximation (dropping a constant term)

$$R(A, b, C, d) = \lambda \frac{n}{12} \left\| \begin{bmatrix} A \\ C \end{bmatrix} \right\|_F^2.$$

This exactly the traditional ridge or quadratic regularizer on the model coefficients, not including the offset.

4.3.2 Feature selection

The regularizer

$$R(A, b, C, d) = \lambda \sum_{i=1}^{p} \|(a_i, c_i)\|_2,$$

where a_i and c_i are the ith columns of A and C, is the sum of the norms of the first p columns of P. This is a well-known sparsifying regularizer, that tends to give coefficients with $(a_i, c_i) = 0$, for many values of i (Meier et al. 2008). This means that the feature entry x_i is not used in the model.

4.3.3 Dual norm regularization

The total number of parameters in our model, given by (5), can be quite large if n is moderate or p is large. An interesting regularizer that leads to a more interpretable covariance predictor can be obtained with dual norm regularization (also called trace or nuclear norm regularization),

$$\lambda \left\| \begin{bmatrix} A \\ C \end{bmatrix} \right\|_*,$$

where $\|\cdot\|_*$ is the dual of the ℓ_2 norm of a matrix, *i.e.*, the sum of the singular values, and $\lambda > 0$ is a hyper-parameter. This regularizer is well known to encourage its argument to be low rank (Vandenberghe and Boyd 1996; Recht et al. 2010).

When $\begin{bmatrix} A \\ C \end{bmatrix}$ is (say) rank r, it can be expressed as the product of two smaller matrices,

$$\begin{bmatrix} A \\ C \end{bmatrix} = UV,$$

where $U \in \mathbf{R}^{n+k \times r}$ and $V \in \mathbf{R}^{r \times p}$. For notational convenience, we let $L^i = \mathbf{mat}(U_i)$, where U_i is the ith column of U and $\mathbf{mat} : \mathbf{R}^{n+k} \to \mathcal{L}$ takes the diagonal and strictly lower triangular entries and gives the corresponding lower triangular matrix in $\mathcal{L}$. We also let $L^0 = \mathbf{mat}(b, d)$. Finally, we let V_i denote the ith row of V.

With this low rank coefficient matrix, the process of prediction can be broken down into two simple steps. We first compute r latent factors $l_i = V_i^T x_i$, $i = 1, \ldots, r$, which are linear in the features. Then $L(x)$ is a sum of $L^0, \ldots, L^r$, weighted by $l_1, \ldots, l_r$,

$$L(x) = L^0 + \sum_{i=1}^{r} l_i L^i.$$

Thus our whitener $L(x)$ is always a linear combination of $L^0, \ldots, L^r$.

4.4 Ordering and permutation

The ordering of the entries in the data y matters. That is, if we fit a model $\hat{\Sigma}_1$ with training data y_i, then fit another model $\hat{\Sigma}_2$ to Qy_i, where Q is a permutation matrix, we generally do not have $\hat{\Sigma}_1(x) = Q^T \hat{\Sigma}_2(x) Q$. This was noted previously in (Sect. 2.2.4, Pourahmadi 2011), where the author states that "the factors of the Cholesky decomposition are dependent on the *order* in which the variables appear in the random vector y_i". This has been noted as a pitfall of Cholesky-based approaches and can lead to significant differences in forecast performance (Heiden 2015), although on problems with real data we have not observed large differences.

This dependence of the prediction model on the ordering of the entries of y is unattractive, at least theoretically. It also immediately raises the question of how to choose a good ordering for the entries of y. We have observed only small differences in the performance of covariance predictors obtained by permuting the entries of y, so perhaps this is not an issue in practice. A reasonable approach is to order the entries in such a way that correlated entries (say, under a base constant model) are near each other (Rothman et al. 2010). But we consider the question of how to order the variables in a regression whitener to be an open question.

There are a number of simple practical ways to deal with this issue. One is to fit a number of models with different orderings of y, and choose the model with the best out of sample likelihood, just as we might do with regularization. In this case we are treating the ordering as a hyper-parameter.

A practical method to obtain a model that is at least approximately invariant under ordering of the entries of y is to fit a number of models $\hat{\Sigma}_1, \ldots, \hat{\Sigma}_K$ that using different orderings, and then to fuse the models via

$$\hat{\Sigma}(x) = \left(\frac{1}{K} \sum_{i=1}^{K} \hat{\Sigma}_i(x)^{-1} \right)^{-1}.$$

Finally, we note that a permutation can be thought of as a very simple whitener in an iterated whitener. It evidently does not whiten the data, but when iterated whitening is done, the permutation can affect the performance of downstream whiteners, such as our regression whitener, that depends on the ordering of the entries of y.

4.5 Implementation

Many methods can be used to solve the convex optimization problem (7). Here we describe some good choices, which are used in our implementation.

4.5.1 L-BFGS

We have observed that with reasonably chosen regularization, the constraint $\|A\|_{\mathrm{row},1} \leq b - \epsilon$ is rarely active at the solution of (7). This suggests that we ignore the constraint, solve the problem, and check at the end if it is active. When the regularizer R is differentiable, the limited-memory Broyden Fletcher Golbfarb Shanno (L-BFGS) method is well suited to solving this problem, after eliminating L_i. The gradients of the objective with respect to (A, b, C, d) are straightforward to work out.

4.5.2 L-BFGS-B formulation

We can use L-BFGS-B (L-BFGS with box constraints) (Liu and Nocedal 1989) to efficiently solve the constrained problem (7), when R is differentiable. We reformulate it as the smooth box-constrained problem

$$\begin{array}{ll}
\text{maximize} & (1/N) \sum_{i=1}^{N} \left(\mathbf{1}^T \log(\mathbf{diag}(L_i)) - (1/2)\|L_i^T y_i\|_2^2 \right) - R(A, b, C, d) \\
\text{subject to} & \mathbf{diag}(L_i) = (A_+ - A_-)x_i + (A_+ + A_-)\mathbf{1} + \epsilon + b_+, \quad i = 1, \ldots, N, \\
& \mathbf{offdiag}(L_i) = Cx_i + d, \quad i = 1, \ldots, N, \\
& A_+ \geq 0, \quad A_- \geq 0, \quad b_+ \geq 0, \\
& A = A_+ - A_, \quad b = (A_+ + A_-)\mathbf{1} + \epsilon + b_+,
\end{array}$$

with variables A_+, A_-, b_+, C, d. Here we have split A into its positive and negative parts, and take $b = (A_+ + A_-)\mathbf{1} + \epsilon + b_+$.

4.5.3 Implementation

We have developed a Python-based object-oriented implementation of the ideas described in this paper, which is freely available online at

`www.github.com/cvxgrp/covpred`.

The only dependencies are `numpy` and `scipy`, and we use `scipy`'s built-in LBFGS-B implementation. The central object in the package is the `Whitener` class, which has three methods: `fit`, `whiten`, and `score`. The `fit` method takes a training dataset given as `numpy` matrices, and fits the parameters of the whitener. The `whiten` method takes a dataset and returns a whitened version of the outcome as well as $L(x_i)$ and $\hat{\Sigma}(x_i)$ for each element of the dataset. The `score` method computes the log-likelihood of a dataset using the whitener. The current implementation includes the following `Whiteners`:

- `ConstantWhitener`, a constant Σ.
- `DiagonalWhitener`, as described in (2).
- `SMAWhitener` and `EWMAWhitener`, as described in Sect. 3.5.
- `RegressionWhitener`, described in Sect. 4.
- `PermutationWhitener`, permutes the entries in y given a permutation.
- `IteratedWhitener`, described in Sect. 2, takes a list of whiteners, and applies them one by one.

These take arguments as appropriate, *e.g.*, the memory for the SMA whitener, and the choice of regularization for the regression whitener. The examples we present later were implemented using this package, with the code available in the `examples` folder of the package linked above.

5 Variations and extensions

Here we list some variations on and extensions of the methods described above.

5.1 Multiple outcomes

Each data record has the feature vector x and a *set* of outcomes, possibly of varying cardinality. (This reduces to our formulation when there is always just one outcome per record.) This is readily handled by simply replicating the data for each of the outcomes. We transform the single record $x, y_1, \ldots, y_q$ into q records of the form $(x, y_1), \ldots, (x, y_q)$. The methods described above can then be applied. If q can be large compared to n, it might be more efficient to transform the data to outer products, *i.e.*, replace the multiple outcomes $y_1, \ldots, y_q$ into $Y = \sum_{i=1}^{q} y_i y_i^T$.

5.2 Handling a nonzero mean

One simple extension is when the outcome vector y has a nonzero mean, and we model its distribution, conditioned on x, as $y \mid x \sim \mathcal{N}(\hat{\mu}(x), \hat{\Sigma}(x))$. One simple approach is sequential: we first fit a model $\hat{\mu}(x)$ of $y \mid x$, for example by regression, subtract it from y to create the prediction residuals or regression errors, and then fit a covariance prediction to the residuals.

5.2.1 Joint prediction of conditional mean and covariance

It is also possible to handle the mean and covariance jointly, using convex optimization. With a nonzero mean $\hat{\mu}(x)$, the log-likelihood (1) becomes

$$-(n/2)\log(2\pi) + \sum_{j=1}^{n} \log L(x)_{jj} - (1/2)\|L(x)^T(y - \hat{\mu}(x))\|_2^2,$$

where, as above, $L(x) = \mathbf{chol}(\hat{\Sigma}(x)^{-1})$. This is concave in $L(x)$ and in $\hat{\mu}(x)$, but not jointly.

A change of variables, however, results in a jointly concave log-likelihood. Changing the mean estimate variable $\hat{\mu}(x)$ to $\nu(x) = L(x)^T\hat{\mu}(x)$, we obtain the log-likelihood function

$$-(n/2)\log(2\pi) + \sum_{j=1}^{n} \log L(x)_{jj} - (1/2)\|L(x)^T y - \hat{\nu}(x)\|_2^2,$$

which is jointly concave in $L(x)$ and $\nu(x)$. We reconstruct the prediction of the mean and covariance of y given x as

$$\hat{\mu}(x) = L(x)^{-T}\nu(x), \qquad \hat{\Sigma}(x) = L(x)^{-T}L(x)^{-1}.$$

This trick is similar to, but not the same as, parametrizing a Gaussian using the natural parameters in the exponential form, $(\Sigma^{-1}, \Sigma^{-1}\mu)$, which results in a jointly concave log-likelihood function. Our parametrization replaces the precision matrix Σ^{-1} with its Cholesky factor L, and uses the parameters

$$(L, \nu) = (\mathbf{chol}(\Sigma^{-1}), \mathbf{chol}(\Sigma^{-1})^T\mu),$$

but we still obtain a concave log-likelihood function.

To carry out joint mean and covariance prediction with the regression whitener, we introduce two additional predictor parameters $E \in \mathbf{R}^{n\times p}$ and $f \in \mathbf{R}^n$, with $\nu(x) = Ex + f$. Maximizing the log-likelihood minus a convex regularizer on (A, b, C, d, E, f) is a convex problem, solved using the same methods as when the mean of y is presumed to be zero.

We note that while our prediction $\nu(x)$ is an affine function of x, our prediction of the mean $\hat{\mu}(x)$ is a nonlinear function of x.

5.3 Structured covariance

A few constraints on the inverse covariance matrix can be expressed as convex constraints on L, and therefore directly handled; others can be handled heuristically. As an example, consider the constraint that $\hat{\Sigma}^{-1}$ be banded, say, tri-diagonal. This is equivalent to $L(x)$ having the same bandwidth (and also, of course, being lower triangular), which in turn translates to rows of C and d corresponding to entries in L outside the band being zero. This can be exactly handled by convex optimization.

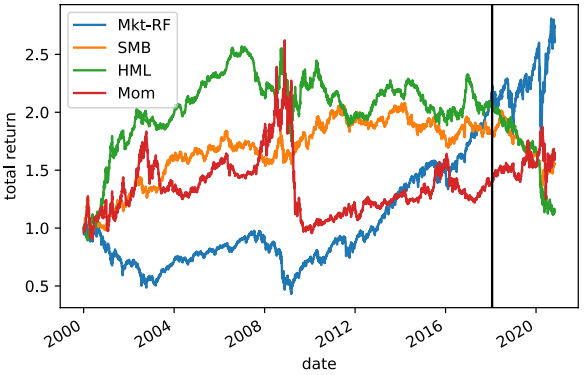

Fig. 1 The cumulative return of the four factors from 2000 to 2020. The vertical black line denotes the split between the train and test samples

Sparsity of $\hat{\Sigma}^{-1}$ (which corresponds to many pairs of the components of y being independent, conditioned on all others) can be approximately handled by insisting that $L(x)$ be very sparse, which in turn can be heuristically handled by using a regularization that encourages row sparsity in C and d, for example a sum of row norms. Similar regularization functions have been used in the context of regularizing covariance predictors (Huang et al. 2006).

6 Example: financial factor returns

In this section we illustrate the methods described above on a financial vector time series, where the outcome consists of four daily returns, and the feature vector is constructed from a volatility index as well as past realized volatilities.

6.1 Outcome and features

6.1.1 Outcome

We take $n = 4$, with y_i the daily returns of four Fama-French factors (Fama and French 1992):

- *Mkt-Rf*, the market-cap weighted return of US equities minus the risk free rate,
- *SMB*, the return of a portfolio that is long small stocks and short big stocks,
- *HML*, the return of a portfolio that is long value stocks and short growth stocks, and
- *Momentum*, the return of a portfolio that is long high momentum stocks and short low (or negative) momentum stocks.

The daily returns have small enough means that they can be ignored.

Our dataset runs from 2000 to 2020. We split the dataset into a training dataset from 2000 to 2018 (4541 samples) and a test dataset from 2018 to 2020 (700

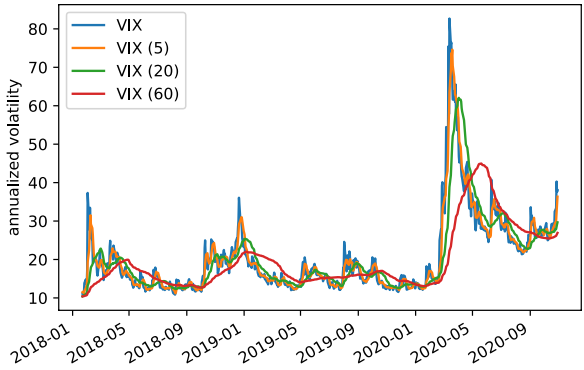

Fig. 2 The four VIX features over the test set

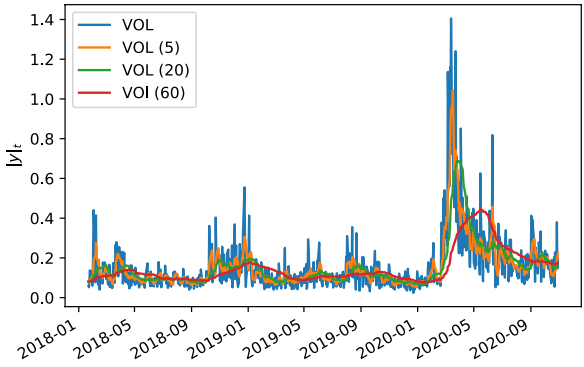

Fig. 3 The four VOL features over the test set

samples). The cumulative return of these four factors (*i.e.*, $\prod_{\tau=1}^{t}(1+(y_\tau)_i)$) from 2000 to 2020 is shown in Fig. 1.

6.1.2 VIX features

Our covariance models use several features derived from the CBOE volatility index (VIX), a market-derived measure of expected 30-day volatility in the US stock market. We use the previous close of VIX as our raw feature. We perform a quantile transform of VIX based on the training dataset, mapping it into $[-1, 1]$ as described in Sect. 4. We will also use 5, 20, and 60 day trailing averages of VIX (which correspond to one week, around one month, and around one quarter). These features are also quantile transformed to $[-1, 1]$. These four features are shown, before quantile transformation, in Fig. 2.

6.1.3 VOL features

We use several features derived from previous realized returns, which measure volatility. One is the sum of the absolute daily returns of the four factors over the previous day, *i.e.*, $\|y_{i-1}\|_1$. We also use trailing 5, 20, and 60 day averages of this quantity.

Table 1 Performance of seven covariance predictors on train and test sets

Predictor	Train log-likelihood	Test log-likelihood
Constant	13.60	12.18
SMA (50)	14.81	13.59
VIX	14.37	13.23
TR-VIX	14.40	13.32
TR-VIX-VOL	14.64	13.48
SMA, then TR-VIX-VOL	14.87	13.78
TR-VIX-VOL, then SMA	15.03	14.10

These four features are each quantile transformed and mapped into $[-1, 1]$. These four features are shown in Fig. 3, before quantile transformation.

6.2 Covariance predictors

We experiment with seven covariance prediction methods, organized into three groups.

6.2.1 Simple predictors

- *Constant* Fit a single covariance matrix to the training set.
- *SMA* We use memory $M = 50$, which achieved the highest log-likelihood on the training set.

6.2.2 Regression whitener predictors

These predictors are based off the whitener regression approach described in Sect. 4; we use the regularization function

$$\lambda_1(\|A\|_F^2 + \|C\|_F^2) + \lambda_2(\|b - \mathbf{1}\|_2^2 + \|d\|_2^2),$$

for $\lambda_1, \lambda_2 > 0$. The hyper-parameters λ_1, λ_2 are selected via a coarse grid search. In all cases, we use $\epsilon = 10^{-6}$.

- *VIX.* A regression whitener predictor with one feature, VIX. We use $\lambda_1 = \lambda_2 = 0$.
- *TR-VIX.* A regression whitener predictor with four features: VIX, and 5/20/60-day trailing averages of VIX. We use $\lambda_1 = 10^{-5}$ and $\lambda_2 = 0$.
- *TR-VIX-VOL.* A whitener regression predictor with eight features: VIX and 5/20/60-day trailing averages of VIX, and also $\|y_{i-1}\|_1$, and 5/20/60 day trailing averages. We use $\lambda_1 = 10^{-5}$ and $\lambda_2 = 0$.

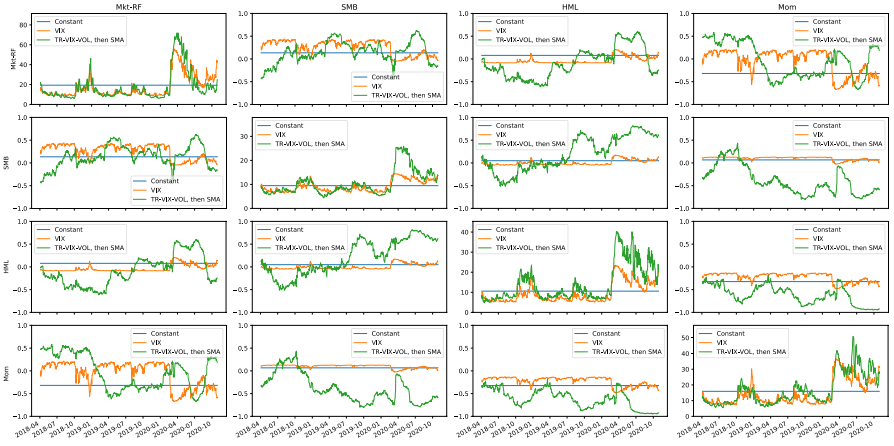

Fig. 4 Predicted annualized volatilities (on the diagonal) and correlations (on the off-diagonal) of the four factors from some of the methods

6.2.3 Iterated predictors

- *SMA, then TR-VIX-VOL.* We first whiten with SMA with memory 50, then with regression using TR-VIX-VOL. For the regression predictor, we use $\lambda_1 = 10^{-5}$ and $\lambda_2 = 10^4$.
- *TR-VIX-VOL, then SMA.* We first whiten with a regression with TR-VIX-VOL, then with SMA, with memory 50. For the regression predictor, we use $\lambda_1 = 10^{-5}$ and $\lambda_2 = 0$.

6.3 Results

The train and test log-likelihood of the seven covariance predictors are reported in Table 1. We can see that a simple moving average with memory 50 does well, in fact, better than the basic predictors based on whitener regressions of VIX and features derived from VIX. However, the iterated whitening predictors, SMA followed by TR-VIX, does somewhat better, with TR-VIX followed by SMA doing the best. This predictor gives an increase in likelihood over the SMA predictor of $\exp(14.1 - 13.59) = 1.67$, *i.e.*, a 67% lift.

6.3.1 Predicted covariances

Figure 4 shows the predicted volatilities and correlations of three of the covariance predictors over the test set, with the volatilities given in annualized percent, *i.e.*, $100\sqrt{250\Sigma_{ii}}$. (The number of trading days in one year is around 250.) The ones that achieve high test log-likelihood vary considerably, with several correlations changing sign over the test period.

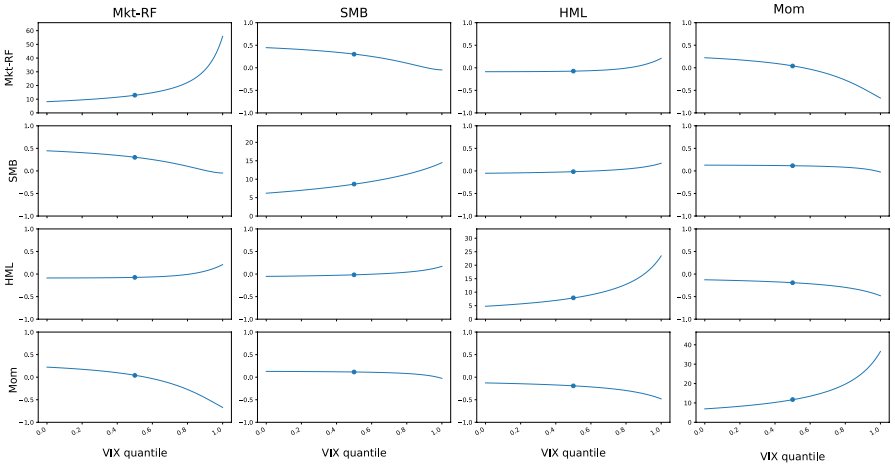

Fig. 5 Predicted annualized volatilities (on the diagonal) and correlations (off-diagonal) of the four factors versus VIX quantile, for the VIX regression covariance predictor. The dot represents the volatility or correlation when VIX is at its median

6.3.2 Effect of ordering outcome components

Five of the predictors used in this example include the regression whitener, which as mentioned above depends on the ordering of the components in y. In each case, we tried all $4! = 24$ permutations (using the permutation whitener class), and found only negligible differences among them. For all 24 permutations, the TR-VIX-VOL, then SMA predictor, achieved the top test performance log-likelihood, with test log-likelihood ranging from 14.072 to 14.105.

6.3.3 The simple VIX regression predictor

The simple VIX regression model is readily interpretable. Our predictor is

$$\begin{bmatrix} 129.4 & 0 & 0 & 0 \\ -58.3 & 184.2 & 0 & 0 \\ 14.8 & -1.0 & 205.1 & 0 \\ 0.8 & -15.9 & 26.4 & 135.7 \end{bmatrix} + x \begin{bmatrix} -90.5 & 0 & 0 & 0 \\ 64.6 & -73.2 & 0 & 0 \\ -1.8 & -13.1 & -128.2 & 0 \\ 43.1 & 12.7 & -2.7 & -92.5 \end{bmatrix},$$

where $x \in [-1, 1]$ is the (transformed) quantile of VIX. The lefthand matrix is the whitener when $x = 0$, *i.e.*, VIX takes its median value. The righthand matrix shows how the whitener changes with x. For example, as x varies over its range $[-1, 1]$, $(L)_{11}$ varies over the range [38.9, 220.0], a factor of around of 5.7. We can easily understand how the predicted covariance changes as x (the quantilized shifted VIX) varies. Figure 5 shows the predicted volatilities (on the diagonal) and correlations (on the off-diagonals) as the VIX feature ranges over $[-1, 1]$. We see that as VIX increases, all the predicted volatilities increase. But we can also see that VIX has an

Table 2 Train and test log-likelihood of the 1, 20, 60, and 250-day predictors

Days	Train log-likelihood	Test log-likelihood
1	14.36	13.29
20	14.24	12.83
60	14.09	11.93
250	13.91	11.75

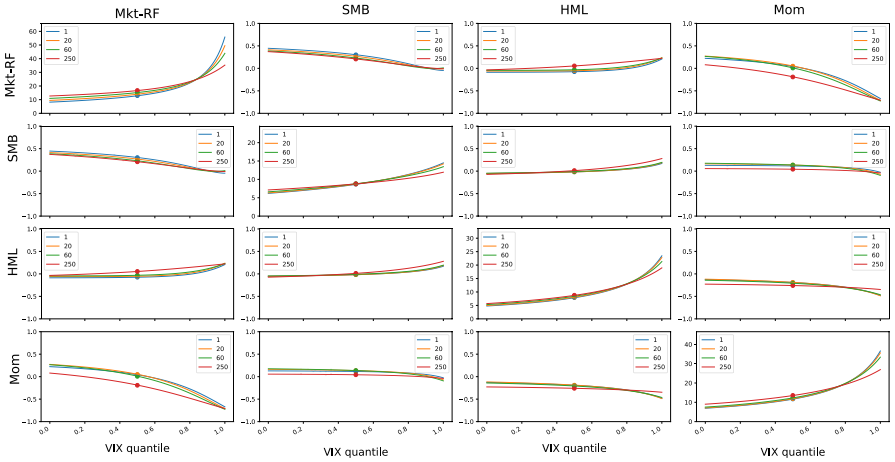

Fig. 6 Predicted annualized volatilities (on the diagonal) and correlations (off-diagonal) of the four factors versus VIX quantile, for the VIX regression covariance predictor for four horizons: 1, 20, 60, and 250 days. The dot represents the predicted volatility or correlation when VIX is at its median

effect on the correlations. For example, when VIX is low, the momentum factor is positively correlated with the market factor; when VIX is high, the momentum factor becomes negatively correlated with the market factor, according to this model.

6.4 Multi-day covariance predictions

The covariance predictors above predict the covariance of the return over the next trading day, *i.e.*, y_i. In this section we form covariance predictors for the next 1, 20, 60, and 250 trading days. (The last three correspond to around one calendar month, quarter, and year.) As mentioned in Sect. 5.1, this is easily done with the same model, by replicating each data point. For example, to predict a covariance matrix for the next 5 days, we take the data record (x_i, y_i) and form five data records,

$$(x_i, y_i),\ (x_i, y_{i+1}),\ \ \ldots,\ (x_i, y_{i+4}),$$

and then use our method to predict the covariance.

We form multi-day covariance predictions over the next 1, 20, 60, and 250 training days for the VIX regression predictor. We report the train and test log-likelihoods

of each of these predictors in Table 2. As expected, the log-likelihood decreases as the number of days ahead we need to predict increases. Figure 5 shows the predicted volatilities (on the diagonal) and correlations (on the off-diagonals) as the VIX feature ranges over $[-1, 1]$ for the 1, 20, 60, and 250-day predictors. We see that the predicted volatility for the market factor over the next day is much more sensitive to VIX than the predicted volatility over the next 250 days. This suggests that volatility is mean-reverting, *i.e.*, over the long run, volatility tends to return to its mean value. We also observe a similar phenomenon with the correlations, although it is less pronounced; for example, the correlation between the HML and momentum factor can go from -0.2 to -0.5 based on VIX on up to 60-day horizons, but stays more or less constant at -0.3 over the 250-day horizon (Fig. 6).

7 Example: machine learning residuals

In this section we present an example where the predicted covariance is of the prediction error or residuals of a point predictor.

7.1 Outcome and features

7.1.1 Dataset

We consider the "Communities and Crime" dataset from the UCI machine learning repository [43,44,45] (Redmond and Baveja 2002; Asuncion and Newman 2007). The dataset consists of 128 attributes of 1994 communities within the United States. These attributes describe the demographics of the community, as well as the socioeconomic conditions and crime statistics. We removed the attributes that are categorical or have missing values, leaving 100 attributes. All attributes came normalized in the range [0, 1]. We randomly split the dataset into a 1495-sample training dataset and a 499-sample test dataset.

7.1.2 Outcome

We choose the following two attributes to be the outcome:

- *agePct65up*. The fraction of the population age 65 and up.
- *pctWSocSec*. The fraction of the population that has social security income.

(These were intentionally picked because they have non-trivial correlation.) We map each of these two attributes to have unit normal marginals on the training set by quantilizing each feature and then applying the inverse CDF of the unit normal.

7.1.3 Features

We use the remaining 98 attributes as the features. We use a quantile transformation for each of these using the training set, and map the resulting features in the train and test set to $[-1, 1]$.

7.2 Regression residual covariance predictors

In our first example we take the simple approach mentioned in Sect. 5.2, where we first form a predictor of the mean, and then form a model of the covariance of the residuals.

7.2.1 Ridge regression model

We fit a ridge regression model to predict the two output attributes from the 98 input attributes, using cross validation on the training set to select the regularization parameter. The root mean squared error (RMSE) of this model on the training set was 0.352 and on the test set was 0.359.

7.2.2 Regression residuals

We use the residuals from the regression model as y_i. That is, if y_i^{true} is the true outcome, and our regression model predicts $\hat{y}_i$, then we let $y_i = y_i^{\text{true}} - \hat{y}_i$. Our goal is to model the covariance of the residuals y_i using x_i as features.

We experiment with seven covariance prediction methods, organized into three groups.

7.2.3 Constant predictor

Fit a single covariance matrix to the training set. This covariance matrix was

$$\Sigma = \begin{bmatrix} 0.14 & 0.08 \\ 0.08 & 0.11 \end{bmatrix}.$$

7.2.4 Regression whitener predictors

Hyper-parameters were selected via a very coarse grid search.

- *Diagonal.* The diagonal predictor in (2) with the regularization function $(0.1)\|A\|_F^2$.
- *Regression.* The regression whitener predictor in Sect. 4 with $\epsilon = 10^{-2}$ and the regularization function $(0.01)(\|A\|_F^2 + \|C\|_F^2)$.

Table 3 Performance of seven covariance predictors on train and test sets

Predictor	Train log-likelihood	Test log-likelihood
Constant	−0.47	−0.44
Diagonal	−0.37	−0.55
Regression	−0.05	−0.14
Constant, then diagonal	−0.45	−0.69
Diagonal, then constant	0.00	−0.18
Constant, then regression	−0.26	−0.33
Regression, then constant	0.01	−0.11

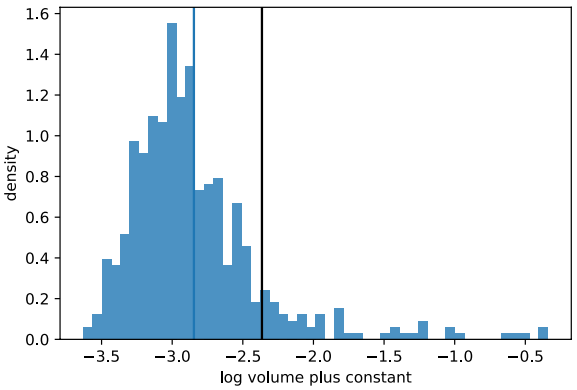

Fig. 7 Log volume of the regression, then constant predictor over the test set. The blue vertical line is the average log volume, and the black vertical line is the log volume of the constant predictor

7.2.5 Iterated predictors

Hyper-parameters were selected via a very coarse grid search.

- *Constant, then diagonal.* The constant predictor, followed by a diagonal predictor with the regularization function $(0.1)\|A\|_F^2$.
- *Diagonal, then constant.* The diagonal predictor with the regularization function $(0.1)\|A\|_F^2$, followed by a constant predictor.
- *Constant, then regression.* The constant predictor, followed by a whitener regression predictor with $\epsilon = 10^{-2}$ and the regularization function $\|A\|_F^2 + \|C\|_F^2 + \|b - 1\|_2^2 + \|d\|_2^2$.
- *Regression, then constant.* The whitener regression predictor with $\epsilon = 10^{-2}$ and the regularization function $(0.1)(\|A\|_F^2 + \|C\|_F^2)$, followed by a constant predictor.

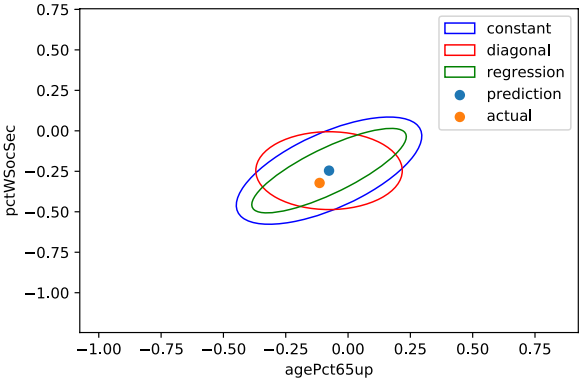

Fig. 8 Confidence ellipsoid of a test sample for three covariance predictors, and the actual outcome

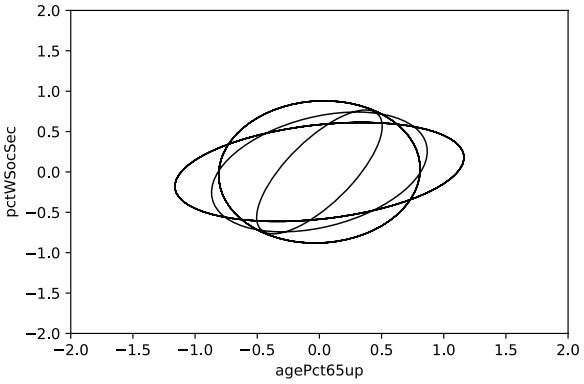

Fig. 9 Some extreme confidence ellipsoids on the test set for the 'regression, then constant' predictor

7.3 Results

The train and test log-likelihood of the seven covariance predictors are reported in Table 3. We can see that the diagonal predictor does the worst, likely because the diagonal predictor fails to model the substantial correlation between the outcomes. The whitener regression predictor does much better than the constant predictor, with a lift of 35% in likelihood. The best predictor was the regression whitener, then constant predictor, with a lift of 39% in likelihood over the constant predictor.

7.3.1 Covariance variation

The regression, then constant predictor predicts a different covariance matrix for each residual, which varies significantly over the test dataset. The standard deviation of the first component varies over the range [0.19, 1.16], and the standard deviation of the second component varies over the range [0.18, 0.88]. The correlation between the two components varies over the range $[-0.045, 0.908]$, *i.e.*, from slightly negatively correlated to strongly correlated.

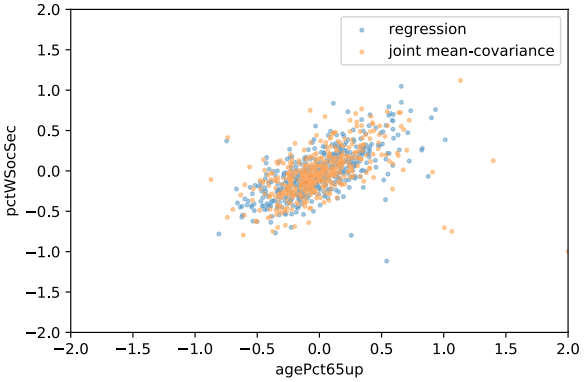

Fig. 10 Test set residuals for the regression predictor and joint mean-covariance predictor

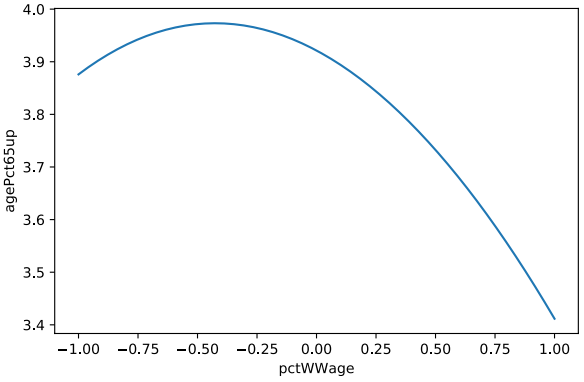

Fig. 11 The nonlinear effect of a particular feature on the mean prediction in the joint mean-covariance model

7.3.2 Volume of the confidence ellipsoids

The confidence regions of a multivariate Gaussian are ellipsoids, with volume proportional to $\det(\Sigma)^{1/2}$. In Fig. 7 we plot the log volume (which here is area since $n = 2$) plus a constant of the 'regression, then constant' predictor over the test set, along with the equivalent log volume of the constant predictor. Most of the time, the volume of the confidence ellipsoid predicted by the best predictor is smaller than the constant predictor. Indeed, on average, the confidence ellipsoid occupies 62% of the area.

7.3.3 Visualization of confidence ellipsoids

In Fig. 8 we plot the one-σ confidence ellipsoids of predictions, on a particular sample from the test dataset. We see that the area of the constant ellipsoid is much larger than the two other predictors, and that the diagonal predictor does not predict any correlation between the outcomes. However, the regression predictor does predict a correlation, and significantly less standard deviation in both outcomes. For this particular example, the regression predictor confidence region is 51% the area of the constant predictor. In

Fig. 9 we visualize some extreme one-σ confidence ellipsoids of the 'regression, then constant' predictor on the test set, demonstrating how much they vary.

7.4 Joint mean-covariance prediction

In this section we perform joint prediction of the conditional mean and covariance as described in Sect. 5.2. We solve the convex problem

$$\begin{array}{ll}\text{maximize} & (1/N)\sum_{i=1}^{N}\left(\sum_{j=1}^{n}\log(L_i)_{jj} - (1/2)\|L_i^T y_i - \nu_i\|_2^2\right) - (0.1)(\|A\|_F^2 + \|C\|_F^2) \\ \text{subject to} & \mathbf{diag}(L_i) = Ax_i + b, \quad i = 1, \ldots, N, \\ & \mathbf{offdiag}(L_i) = Cx_i + d, \quad i = 1, \ldots, N, \\ & \nu_i = Ex_i + f, \quad i = 1, \ldots, N, \\ & \|A\|_{\text{row},1} \leq b - \epsilon, \end{array} \tag{8}$$

with variables (A, b, C, d, E, f) using L-BFGS-B.

By jointly predicting the conditional mean and covariance, we actually achieve a better test RMSE than predicting just the mean. The train RMSE of this model was 0.273 and the test MSE was 0.331, whereas the RMSE of the ridge regression model was 0.352 on the training set and 0.359 on the test set. Thus, jointly modeling the mean and covariance results in a 7.8% reduction in RMSE on the test set.

In Fig. 10 we visualize the residuals of the ridge regression model and the joint mean-covariance model on the test set. We observe that the joint mean-covariance residuals seem to be on average closer to the origin. In terms of Gaussian log-likelihood, the ridge regression model with a constant covariance achieves a train log-likelihood of 0.054 and a test log-likelihood of 0.088. In contrast, the joint mean-covariance model achieves a train log-likelihood of 1.132 and a test log-likelihood of 1.049, representing a lift on the test set of 161%.

Recall that the prediction of the mean by the joint mean-covariance model is nonlinear in the input. In Fig. 11 we visualize this effect for a particular test point, by varying just the 'pctWWage' feature from -1 to 1, and visualizing the change in the mean prediction for the 'agePct65up' output. The nonlinearity is evident.

8 Conclusions and future work

Many covariance predictors, ranging from simple to complex, have been developed. Our focus has been on the regression whitener, which has a concave log-likelihood function, so fitting reduces to a convex optimization problem that is readily solved. The regression whitener is also readily interpretable, especially when the number of features is small, or a rank-reducing regularizer results in a low rank coefficient matrix in the predictor. Among other predictors that have been proposed, the only other ones that share the property of having a concave log-likelihood is the diagonal (exponentiated) covariance predictor and the Laplacian regularized stratified predictor.

We observed that covariance predictors can be iterated; our examples show that simple sequences of predictors can indeed yield improved performance. While iterated covariance prediction can yield better covariance predictors, it raises the question of how to choose the sequence of predictors. At this time, we do not know, and can only suggest a trial and error approach. We can hope that this question is at least partially answered by future research.

As other authors have observed, it would be nice to identify an unconstrained parametrization of covariance matrices, *i.e.*, an inverse link mapping that maps all of $\mathbf{R}^p$ onto $\mathbf{S}^n_{++}$. (Our regression whitener requires the constraint $\|x\|_\infty \leq 1$.) One candidate is the matrix exponential, which maps symmetric matrices onto $\mathbf{S}^n_{++}$. Unfortunately, this parametrization results in a log-likelihood that is not concave. As far as we know, the existence of an unconstrained parametrization of covariance matrices, with a concave log-likelihood, is still an open question.

Finally, we mention one more issue that we hope will be addressed in future research. The dependence of the regression whitener on the ordering of the entries of y certainly detracts from its aesthetic and theoretical appeal. In our examples, however, we have obtained similar results with different orderings, suggesting that the dependence on ordering is not a large problem in practice, though it should still be checked. Still, some guidelines as to how to choose the ordering, or otherwise address this issue, would be welcome.

Acknowledgements The authors gratefully acknowledge conversations and discussions about some of the material in this paper with Misha van Beek, Linxi Chen, David Greenberg, Ron Kahn, Trevor Hastie, Rob Tibshirani, Emmanuel Candes, Mykel Kochenderfer, and Jonathan Tuck.

References

Anderson T (1973) Asymptotically efficient estimation of covariance matrices with linear structure. Ann Stat 1(1):135–141

Anscombe J (1961) Examination of residuals. In: Proceedings of the Berkeley symposium on mathematical statistics and probability

Asuncion A, Newman D (2007) UCI machine learning repository

Bilmes J (1998) A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Int Comput Sci Inst 4(510):126

Bollerslev T (1986) Generalized autoregressive conditional heteroskedasticity. J Econom 31(3):307–327

Bollerslev T (1990) Modelling the coherence in short-run nominal exchange rates: a multivariate generalized ARCH model. Rev Econom Stat, pp 498–505

Bollerslev T, Engle R, Wooldridge J (1988) A capital asset pricing model with time-varying covariances. J Polit Econ 96(1):116–131

Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge University Press, Cambridge

Chiu T, Leonard T, Tsui K-W (1996) The matrix-logarithmic covariance model. J Am Stat Assoc 91(433):198–210

Cleveland W, Devlin S (1988) Locally-weighted regression: an approach to regression analysis by local fitting. J Am Stat Assoc 83(403):596–610

Cook D, Weisberg S (1983) Diagnostics for heteroscedasticity in regression. Biometrika 70(1):1–10

Davidian M, Carroll R (1987) Variance function estimation. J Am Stat Assoc 82(400):1079–1091

Dempster A (1972) Covariance selection. Biometrics, pp 157–175

Dorta G, Vicente S, Agapito L, Campbell N, Simpson I (2018) Structured uncertainty prediction networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5477–5485
Engle R (1982) Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. Econom J Econom Soc, pp 987–1007
Engle R, Kroner K (1995) Multivariate simultaneous generalized ARCH. Econom Theory, pp 122–150
Fama E, French K (1992) The cross-section of expected stock returns. J Finance 47(2):427–465
Francq C, Zakoian J-M (2019) GARCH models: structure, statistical inference and financial applications. John Wiley and Sons, New Jersey
Freund Y, Schapire R (1996) Experiments with a new boosting algorithm. In: icml, vol 96, pp 148–156. Citeseer
Friedman J, Hastie T, Tibshirani R (2008) Sparse inverse covariance estimation with the graphical lasso. Biostatistics 9(3):432–441
Grinold R, Kahn R (2000) Active portfolio management. McGraw Hill, New York
Harper D (2009) Exploring the exponentially weighted moving average. Investopedia
Hawkins D, Maboudou-Tchao E (2008) Multivariate exponentially weighted moving covariance matrix. Technometrics 50(2):155–166
Heiden M (2015) Pitfalls of the Cholesky decomposition for forecasting multivariate volatility. Available at SSRN 2686482
Huang J, Liu N, Pourahmadi M, Liu L (2006) Covariance matrix selection and estimation via penalised normal likelihood. Biometrika 93(1):85–98
Liu D, Nocedal J (1989) On the limited memory BFGS method for large scale optimization. Math Program 45(1–3):503–528
Longerstaey J, Spencer M (1996) Riskmetrics – Technical Document. JP Morgan and Reuters
Meier L, Van De Geer S, Bühlmann P (2008) The group lasso for logistic regression. J R Stat Soc Ser B (Stat Methodol) 70(1):53–71
Menchero J, Orr DJ, Wang J (2011) The Barra US equity model (USE4), methodology notes. MSCI Barra
Jianxin P (2021) jmcm: Joint mean-covariance models using 'Armadillo' and S4. R package version 0.2.4
Posthuma P (2019) lmvar: Linear Regression with Non-Constant Variances. R package version 1.5.2
Pourahmadi M (1999) Joint mean-covariance models with applications to longitudinal data: Unconstrained parameterisation. Biometrika 86(3):677–690
Pourahmadi M (2000) Maximum likelihood estimation of generalised linear models for multivariate normal covariance matrix. Biometrika 87(2):425–435
Pourahmadi M (2011) Covariance estimation: The GLM and regularization perspectives. Stat Sci, pp 369–387
Recht B, Fazel M, Parrilo P (2010) Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. SIAM Rev 52(3):471–501
Redmond M, Baveja A (2002) A data-driven software tool for enabling cooperative information sharing among police departments. Eur J Oper Res 141:660–678
Rothman A, Levina E, Zhu J (2010) A new approach to Cholesky-based covariance regularization in high dimensions. Biometrika 97(3):539–550
Rubin D, Thayer D (1982) EM algorithms for ML factor analysis. Psychometrika 47(1):69–76
Liangjun S, Wang X (2017) On time-varying factor models: estimation and testing. J Econom 198(1):84–101
Tuck J, Barratt S, Boyd S (2021) A distributed method for fitting Laplacian regularized stratified models. J Mach Learn Res 22:60–1
Tuck J, Barratt S, Boyd S (2021) Portfolio construction using stratified models. arXiv preprint arXiv:2101.04113
Tuck J, Boyd S (2020) Fitting Laplacian regularized stratified gaussian models. arXiv preprint arXiv:2005.01752
Bureau of the Census US Department of Commerce. Census of Population and Housing 1990 United States: Summary tape file 1a and 3a (computer files)
Bureau of Justice Statistics US Department of Justice (1992) Law enforcement management and administrative statistics (computer file)
Federal Bureau of Investigation US Department of Justice (1995) Crime in the United States (computer file)
Vandenberghe L, Boyd S (1996) Semidefinite programming. SIAM Rev 38(1):49–95

Williams P (1996) Using neural networks to model conditional multivariate densities. Neural Comput 8(4):843–854
Williams P (1999) Matrix logarithm parametrizations for neural network covariance models. Neural Netw 12(2):299–308
Wei W, Pourahmadi M (2003) Nonparametric estimation of large covariance matrices of longitudinal data. Biometrika 90(4):831–844