Volatility Estimation with Functional Gradient Descent for Very High-Dimensional Financial Time Series

Francesco Audrino and Peter Bühlmann
ETH Zürich, Switzerland

August 2001

Abstract

We propose a functional gradient descent algorithm (FGD) for estimating volatility and conditional covariances (given the past) for very high-dimensional financial time series of asset price returns. FGD is a kind of hybrid of nonparametric statistical function estimation and numerical optimization. Our FGD algorithm is computationally feasible in multivariate problems with dozens up to thousands of individual return series. Moreover, we demonstrate on some synthetic and real data-sets with dimensions up to 100, that it yields significantly, much better predictions than more classical approaches such as a constant conditional correlation GARCH-type model. Since our FGD algorithm is constructed from a generic algorithm, the technique can be easily adapted to other problems of learning in very high dimensions.

Heading: High-dimensional volatility estimation

1 Introduction

Returns of asset prices in a portfolio generate a multivariate time series, often in dozens or hundreds of dimensions. Denote by $P_{t,i}$ the price of asset i at time t and its returns by $X_{t,i} = (P_{t,i} - P_{t-1,i})/P_{t-1,i}$. The multivariate time series $\{X_{t,i}; t = 1, \ldots, n, i = 1, \ldots, d\}$ is usually assumed to be stationary and most of the modeling and prediction effort is on the multivariate (squared) volatility

$$V_t = \mathrm{Cov}_{d\times d}(\mathbf{X}_t|\mathcal{F}_{t-1}),\ \mathbf{X}_t = (X_{t,1}, \ldots, X_{t,d})^T, \tag{1.1}$$

where $\mathcal{F}_{t-1}$ denotes the information up to time $t-1$, the σ-algebra generated by $\{\mathbf{X}_s; s \leq t-1\}$. Besides the genuine interest of volatility in finance, V_t is a key quantity because the following model often yields a reasonable approximation,

$$\mathbf{X}_t = \Sigma_t \mathbf{Z}_t, \tag{1.2}$$

where $\Sigma_t \Sigma_t^T = V_t$ and $\mathbf{Z}_t$ are i.i.d. multivariate innovations with uncorrelated components and componentwise variances equal to one. Due to the enormous complexity of V_t as a function of the past $\mathcal{F}_{t-1}$ for d in the hundreds, the problem of predicting such high-dimensional V_t received only little attention, particularly when (nonlinear) methods are used which take cross-dependencies between the time series and auto-dependences from past values into account. Predicting truly high-dimensional volatility in (1.1) raises huge challenges in computational and modeling issues due to the well known curse of dimensionality. Previous work on multivariate volatility models has been given by Bollerslev (1990), Engle et al. (1990), Lin (1992) and Engle and Kroner (1995) in the framework of GARCH-type models, and by Harvey et al. (1994), Aguilar and West (2000) and Chib et al. (1999) within the stochastic volatility (state space model) framework. In the GARCH-type framework, only very simple models (Bollerslev, 1990) are feasible in high dimensions, whereas with stochastic volatility models, only Chib et al. (1999) present an example with dimensionality as large as 40 which is still far lower than the degree of multivariateness we can deal with here and which often occurs in practice.

We propose here a version of functional gradient descent (FGD), a recent technique from the area of machine learning (Breiman, 1999; Mason et al., 1999; Friedman et al., 2000; Friedman, 2001). Our FGD method is based on the likelihood-framework of a-priori very general GARCH-type models. FGD has mainly influenced the thinking around so-called Boosting (Freund and Schapire, 1996) which is a machine learning technique for the classification problem (predicting labels or classes from many explanatory variables). Some important modifications of more standard FGD algorithms are necessary to make the approach successful in the very different field here of high-dimensional financial time series.

It is well known that the multivariate approach is needed in many areas such as risk management and portfolio analysis. For example, the study of a portfolio

$$P_t = \sum_{i=1}^{d} \alpha_{t,i} P_{t,i}$$

with time-changing weights requires the multivariate approach. More generally, the aim is often to estimate the conditional distribution of a "pay-out" function

$$\psi_t(P_{t,1}, \ldots, P_{t,d})$$

given the information $\mathcal{F}_{t-1}$, where ψ_t could also be nonlinear. To do so, it suffices to estimate the distribution of $\mathbf{X}_t$ given $\mathcal{F}_{t-1}$; and this boils mainly down to estimating the volatility matrix V_t when assuming a model as in (1.2) (estimating the innovation distribution of $\mathbf{Z}_t$ is then easily done via estimated residuals).

Mainly for conceptual purposes, we present in section 3 FGD for volatility estimation in univariate time series. The method produces accurate results and is better than classical GARCH prediction. But as argued above, the merit is mainly for very high-dimensional problems where there is virtually no other competitive alternative method; this will be discussed in section 4. Numerical results are illustrated on real and simulated data.

2 The generic functional gradient descent algorithm

We are presenting here the main idea of functional gradient descent (FGD) in the framework of general regression. Consider data $(Y_1, X_1), \ldots, (Y_n, X_n)$, where Y_i is the response and X_i the explanatory variable. For simplicity, we assume here that Y_i is univariate (e.g. taking values in $\mathbb{R}$) and X_i is p-dimensional (e.g. taking values in $\mathbb{R}^p$).

The aim is to estimate a function $F(x)$ such as $F(x) = \mathbb{E}[Y|X = x], F(x) = \mathrm{Var}(Y|X = x)$ or $F(x) = \mathbb{P}[Y = 1|X = x]$ if $Y \in \{0, 1\}$ is binary. The function $F(\cdot)$ can often be represented as the minimizer (in function space) of an expected loss function $\lambda(y, f)$,

$$F(\cdot) = \mathrm{argmin}_F \mathbb{E}[\lambda(Y, F(X))]$$

As an example, $F(x) = \mathbb{E}[Y|X = x]$ can be represented as the minimizer of such an expected loss with $\lambda(y, f) = (y - f)^2$. The functional gradient descent estimate of $F(\cdot)$ is then constructed from a constrained minimization of the empirical risk

$$n^{-1} \sum_{i=1}^{n} \lambda(Y_i, F(X_i)).$$

The constraints require that the solution $\hat{F}(\cdot)$ is an additive expansion of "simple estimates",

$$\hat{F}_M(\cdot) = \sum_{m=0}^{M} \hat{w}_m \hat{f}_m(\cdot). \tag{2.1}$$

The "simple estimates" are given from a base learner $\mathcal{S}$ where

$\mathcal{S}_X(U)(x)$ denotes the predicted value at $x \in \mathbb{R}^p$ from the base learner $\mathcal{S}$,

using the response vector $U = (U_1, \ldots, U_n) \in \mathbb{R}^n$ and explanatory variables $X = X_1, \ldots, X_n \in \mathbb{R}^{pn}$. Typically, $\mathcal{S}_X(U)(x)$ is an estimate of $\mathbb{E}[U_1|X_1 = x]$. It is often constructed from (constrained) least squares fitting

$$\mathcal{S}_X(U)(\cdot) = \mathrm{argmin}_f \sum_{i=1}^{n} (U_i - f(X_i))^2.$$

It could be the fit from a base learner such as a regression tree, a projection pursuit or a neural net.

The constraints in the additive expansion in (2.1) are automatically built in when proceeding with the following generic FGD algorithm, cf. Friedman (2001).

Generic functional gradient descent (FGD)

Step 1 (initialization). Specify the starting function $\hat{F}_0(\cdot)$ and set $m = 1$.

Step 2 (projection of gradient to base learner). Compute the negative gradient

$$U_i = -\frac{\partial \lambda(Y_i, F)}{\partial F}|_{F=\hat{F}_{m-1}(X_i)},\ i = 1, \ldots, n,$$

evaluated at the previous estimate $\hat{F}_{m-1}(\cdot)$ and the data points. Then, fit the negative gradient vector with a base learner $\mathcal{S}$

$$\hat{f}_m(\cdot) = \mathcal{S}_X(U)(\cdot).$$

The vector $(\hat{f}_m(X_1), \ldots, \hat{f}_m(X_n))^T$ can be viewed as a kind of projection of the negative gradient to the base learner.

Step 3 (line search). Perform a one-dimensional optimization for the step-length when up-dating $\hat{F}_{m-1}$ with $\hat{f}_m$,

$$\hat{w}_m = \text{argmin}_w \sum_{i=1}^{n} \lambda(Y_i, \hat{F}_{m-1}(X_i) + w\hat{f}_m(X_i)).$$

Up-date

$$\hat{F}_m(\cdot) = \hat{F}_{m-1}(\cdot) + \hat{w}_m \hat{f}_m(\cdot).$$

Step 4 (iteration). Increase m by one an iterate Steps 2 and 3 until stopping with $m = M$. This produces the FGD estimate

$$\hat{F}_M(\cdot) = \hat{F}_0(\cdot) + \sum_{m=1}^{M} \hat{w}_m \hat{f}_m(\cdot).$$

The stopping value M is chosen to optimize a measure for out-sample prediction.

Remark 1. Initialization in Step 1 was so far believed to be of negligible importance: an initial function often proposed is $\hat{F}_0 \equiv \overline{Y}_n$. But we will see in section 3, from an empirical point of view, that initialization does play an important role in the application of volatility estimation.

Remark 2. The line search in Step 3 guarantees that the empirical risk is monotonely decreasing with every iteration.

Remark 3. Stopping in Step 4 is important. Typically, the algorithm would fit the data perfectly as iterations tend to infinity.

Numerical optimization has advanced to faster converging algorithms than steepest gradient methods. However, in our setting the slow "convergence" (note that we do not iterate until convergence) is very helpful as a device for regularization.

The name "functional gradient" can be justified as follows. We are asking for the "direction" f (in function space) such that $\Lambda(\hat{F}_{m-1} + \varepsilon f)$ most rapidly decreases, for small value of ε, where $\Lambda(F) = n^{-1}\sum_{i=1}^{n}\lambda(Y_i, F(X_i))$. Viewing Λ as a functional on $\mathrm{lin}(\mathcal{F})$, the set of linear combinations of functions in a suitable class of base learners $\mathcal{F}$, the desired direction is the negative functional derivative $-d\Lambda(F,\cdot)$, where

$$d\Lambda(F,x) = \lim_{\varepsilon \searrow 0} \frac{\Lambda(F + \varepsilon \mathbb{I}_{[x]}) - \Lambda(F)}{\varepsilon},\ x \in \mathbb{R}^p,$$

where $\mathbb{I}_{[\cdot]}$ denotes the indicator function. We are restricted to choose $f \in \mathcal{F}$ and cannot choose f as $-d\Lambda(F,x)$. Instead, we search for an f minimizing $\| - d\Lambda(F) - f\|^2$, where $\|g\|^2 = n^{-1}\sum_{i=1}^{n} g(X_i)^2$. This is equivalent to fit the negative gradient vector $(U_1,\ldots,U_n)^T$ in Step 2 with the base learner $\mathcal{S}$ fitted by least squares producing $\hat{f}_m \in \mathcal{F}$.

Why should we use FGD at all? In very high-dimensional settings, particularly in connection with tree-structured base learners (see section 2.1), it is a feasible method aiming to improve the starting function. Bühlmann and Yu (2001) prove a rate-optimality result for FGD with the quadratic loss function λ in one-dimensional function estimation of unknown smoothness. Thus, in such low-dimensional toy problems, FGD achieves some optimality criterion asymptotically. But the theory for very high-dimensional settings or other loss functions λ is not rigorously developed so far. Good empirical performance of FGD, particularly in the area of classification with boosting (Freund and Schapire, 1996; Friedman et al., 2000; Friedman, 2001), and the asymptotic results in univariate function estimation support evidence that FGD is a reasonable strategy. These arguments are further strengthened by the fact that there are not many other methods whose computational costs remains manageable in very high-dimensional, large scale problems.

2.1 Choice of the base learner

The base learner in Step 2 of the functional gradient descent algorithm, producing the additive terms $\hat{f}_m(\cdot)$, obviously determines the FGD estimate $\hat{F}_M(\cdot)$. The base learner should be "weak", i.e. not too complex or high-dimensional, so that FGD would not immediately produce an overfitted estimate with the first iteration. By adding further additive terms with every iteration, we increase complexity or dimensionality of the FGD estimate $\hat{F}_M$. However, this increase in complexity is not linear and performing further iterations typically changes complexity only by "small" amounts; for example, it is shown for certain cases to become exponentially diminishing as FGD iterations grow (Bühlmann and Yu, 2001). As usual, a bias-complexity trade-off is present: the complexity of the base learner and the number of FGD iterations determine the trade-off in a somewhat unusual way, see also Bühlmann and Yu (2001).

Often, decision trees are used as base learners. Particularly in high dimensions, they have the ability to do variable selection by choosing just a few of the explanatory variables for prediction. We will consider here decision trees but also projection pursuit (Friedman and Stuetzle, 1981) as base learners. It is often desirable to make a base learner sufficiently "weak", i.e. of sufficiently low complexity. A simple but effective solution to achieve this is via shrinkage towards zero: the up-date $\hat{w}_m\hat{f}_m(\cdot)$ in Step 3 of the FGD algorithms is replaced by

$$\nu\hat{w}_m\hat{f}_m(\cdot),\ 0 < \nu \leq 1. \tag{2.2}$$

Obviously, this reduces the variance (a complexity measure) by the factor ν^2.

3 Univariate volatility estimation with FGD

Having a univariate time series of observed prices $P_0, P_1, \ldots, P_n$ of an asset, we consider their returns $X_t = (P_t - P_{t-1})/P_{t-1}$, or $X_t = \log(P_t/P_{t-1})$ alternatively, and assume stationarity at least in a suitable time-window. The equi-distant time-spacing $\Delta t = t - (t-1) = 1$ is often one business day. Our working model for the time series of such price returns is

$$X_t = \sigma_t Z_t,\ \sigma_t^2 = F(X_{t-1}, X_{t-2}, \ldots), \tag{3.1}$$

where the innovations Z_t are i.i.d. with $\mathbb{E}[Z_t] = 0,\ \mathrm{Var}(Z_t) = 1$ and Z_t independent from $\{X_s; s < t\}$; the function $F : \mathbb{R}^\infty \to \mathbb{R}^+$ is assumed a priori to be very general. It is later the FGD estimate which constrains $F(\cdot)$ to be of a more particular, but often still general form.

For FGD, we assume that $F : \mathbb{R}^p \to \mathbb{R}^+$ with p finite; but we allow the starting function $\hat{F}_0(\cdot)$ to depend on the whole past of the time series. We choose the loss-function $\lambda(\cdot,\cdot)$ from the maximum-likelihood framework with innovations $Z_t \sim \mathcal{N}(0,1)$:

$$\lambda(y,f) = -\log(f^{-1/2}\varphi(yf^{-1/2})) = \frac{1}{2}\left(\log(f) + \frac{y^2}{f} + \log(2\pi)\right).$$

(Of course, we could drop the $\log(2\pi)$ term which would then represent a simpler, equivalent loss function). This, because with Gaussian innovations in (3.1), the negative log-likelihood (conditional on the first p values) is

$$-\sum_{t=p+1}^{n} \log\left(F(X_{t-p}^{t-1})^{-1/2}\varphi(X_t F(X_{t-p}^{t-1})^{-1/2})\right),$$

where $X_{t-p}^{t-1} = X_{t-1}, \ldots, X_{t-p}$. The partial derivative of the loss-function is

$$\frac{\partial \lambda(y,f)}{\partial f} = (f^{-1} - y^2 f^{-2})/2$$

and the FGD algorithm from section 2 can now be used.

As a starting function, we propose to use the fit from a GARCH(1,1) model (Bollerslev, 1986)

$$\hat{F}_0(X_{t-1}, X_{t-2}, \ldots) = \hat{\alpha}_0 + \hat{\alpha}_1 X_{t-1}^2 + \hat{\beta}\hat{F}_0(X_{t-2}, X_{t-3}, \ldots), \tag{3.2}$$

with parameters estimates $\hat{\alpha}_0, \hat{\alpha}_1, \hat{\beta}$ from parametric maximum-likelihood in the GARCH(1,1) model with Gaussian innovations.

Summarizing, the FGD algorithm for univariate volatility estimation then looks as follows.

FGD for univariate volatility

Step 1 (initialization). Choose the starting function $\hat{F}_0(\cdot)$ from (3.2) and denote by $\hat{F}_0(t) = \hat{F}_0(X_1^{t-1})$. Set $m = 1$.

Step 2 (projection of gradient to base learner). Compute the negative gradient

$$U_t = (X_t^2 \hat{F}_{m-1}(t)^{-2} - \hat{F}_{m-1}(t)^{-1})/2, \ t = p+1, \ldots, n$$

Then, fit the negative gradient vector with a base learner, using always the first p time-lagged predictor variables (i.e. X_{t-p}^{t-1} is the predictor for U_t)

$$\hat{f}_m(\cdot) = \mathcal{S}_X(U)(\cdot).$$

Steps 3 and 4. As in the generic FGD algorithm, generally with shrinkage as in (2.2).

The important issue is that the starting function $\hat{F}_0(\cdot)$ matters a lot for obtaining good volatility estimates. We illustrate this in Figure 3.1 by one simulation from the model (3.3) with sample size $n = 1000$. The out-sample OS-L$_2$ losses with $n_{out} = 1000$ (see section 3.1) are 174.5 (FGD with tree as in the Appendix and constant starting function $\hat{F}_0(x) \equiv S^2$ being the empirical overall variance), 94.6 (classical GARCH(1,1) prediction) and 66.9 (FGD with tree as in the Appendix and starting function from a GARCH(1,1) fit).

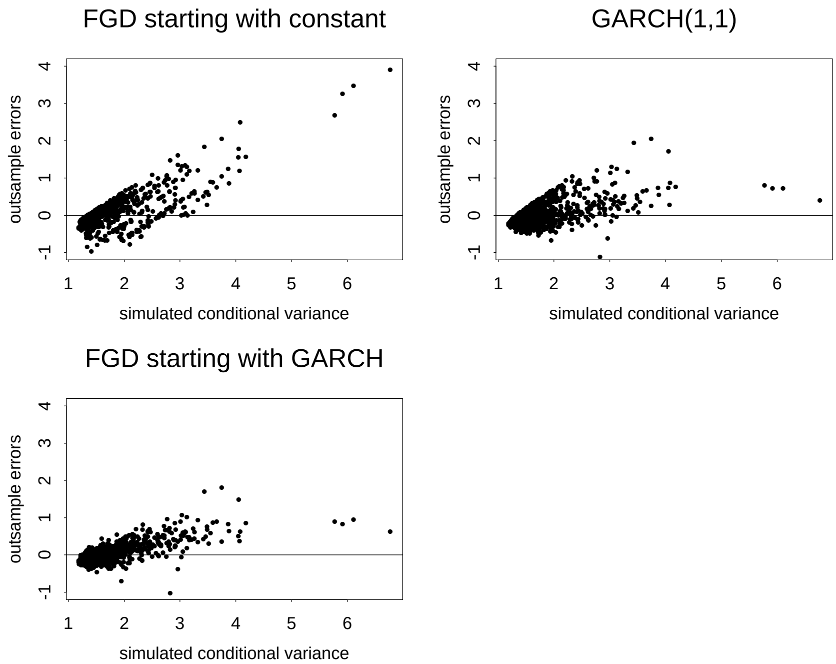

Figure 3.1: Outsample errors $\hat{\sigma}_t^2 - \sigma_t^2$ versus true σ_t^2 in simulated model (3.3). Top left: FDG with trees using constant starting function $\hat{F}_0(x) \equiv S^2$ (empirical marginal variance); Top right: GARCH(1,1) prediction; Bottom: FGD with trees using the GARCH(1,1) fit as starting function.

Also, it is advisable to allow for shrinkage in Step 3. Regarding the base learner $\mathcal{S}$, we have considered regression trees (Breiman at al., 1984) and projection pursuit regression (Friedman and Stuetzle, 1981). With regression trees, the FGD algorithm for univariate volatility estimation can be further modified to achieve marginal improvements: the alternative version is described in the Appendix.

3.1 Numerical results for simulated data

We simulate from the following model,

$$X_t = \sigma_t Z_t,\ \sigma_t^2 = F(X_{t-1}, \sigma_{t-1}^2),$$
$$F(x, \sigma^2) = (0.1 + 0.2\,|x| + 0.9x^2) \cdot (0.8 \exp(-1.5\,|x|\,|\sigma|)) + (0.4x^2 + 0.5\sigma^2)^{3/4} \quad (3.3)$$

where $Z_t \sim \mathcal{N}(0,1)$ is as in (3.1).

For quantifying the goodness of fit, we consider various measures:

$$\text{out-sample negative log-likelihood: } \sum_{t=1}^{n_{out}} \lambda(Y_t, \hat{F}(Y_1^{t-1})), \quad (3.4)$$

$$\text{IS-L}_2 = \sum_{t=1}^{n} |\sigma_t^2 - \hat{F}(X_1^{t-1})|^2 \text{ (in-sample loss)}, \quad (3.5)$$

$$\text{OS-L}_2 = \sum_{t=1}^{n_{out}} |\sigma_t^2 - \hat{F}(Y_1^{t-1})|^2 \text{ (out-sample loss)}, \quad (3.6)$$

where $Y_1, \ldots, Y_{n_{out}}$ are new test observations, independent from but with the same distribution as the data $X_1, \ldots, X_n$. Both, the IS- and OS-L$_2$ statistics are interesting measures in simulations, but we can't evaluate them for real data. The out-sample negative log-likelihood is a more generally applicable measure for out-sample performance. Table 3.1

Model	Performance measure		
	OS −log-likelihood	IS-L$_2$	OS-L$_2$
GARCH(1,1)	1656.363	119.169	111.478
FGD with tree using $p = 1,\ \nu = 0.1,\ L = 3$	1653.434	98.597	91.223
FGD with tree using $p = 2,\ \nu = 0.1,\ L = 5$	1653.534	104.257	96.243
FGD with PPR using $p = 2,\ \nu = 0.1,\ S = 3$	1654.296	114.651	104.932
FGD with PPR using $p = 2,\ \nu = 0.05,\ S = 2$	1654.672	112.467	104.164

Table 3.1: Goodness of fit measures (on average) for fifty simulations from model (3.3) with sample size $n = 1000$. Out-sample performances OS-L$_2$ and negative log-likelihood as in (3.4)-(3.6) are evaluated with test-set of size $n_{out} = 1000$. Notation: number of lagged values (p), shrinkage factor (ν), number of terminal nodes (L) and number of ridge functions (S).

shows the result for 50 independent realizations from model (3.3). Sample size is $n = 1000$ and test-set size is $n_{out} = 1000$. For this case, FGD with trees as in the Appendix is better than FGD with projection pursuit, and both FGD techniques outperform the predictions from classical GARCH(1,1). The differences in the out-sample log-likelihood are small despite that the actual differences in volatility are substantial. This phenomenon is well known and occurs because the out-sample log-likelihood measures quality for predicting *future returns* and not future volatilities; the former is much more noisy than the latter.

Thus, similar out-sample log-likelihoods (or other prediction losses) with different methods are not implying that the methods are similar in terms of the differences between estimated and true volatility, as observed in Table 3.1.

For real data, it is harder to detect differences because appropriate prediction measures like the out-sample negative log-likelihood is affected by a large noise component, as mentioned above. We will show in section 4.3.2 how testing can be used to judge whether out-sample log-likelihoods (or other losses) are significantly different, even when they appear to be similar.

4 Volatility estimation for high multivariate time series

In the multivariate set-up, we have time series of asset prices $\{P_{t,i}; t = 0, 1, \ldots, n, i = 1, \ldots, d\}$. Their returns are defined as

$$X_{t,i} = (P_{t,i} - P_{t-1,i})/P_{t-1,i},\ t = 1, \ldots, n.$$

As mentioned already in section 1, the challenging problem is prediction of the multivariate volatility matrix

$$V_t = \mathrm{Cov}_{d\times d}(\mathbf{X}_t|\mathcal{F}_{t-1}),\ \mathbf{X}_t = (X_{t,1}, \ldots, X_{t,d})^T$$

in dimensions in the hundreds. FGD becomes a powerful strategy to construct computable and good predictions for V_t.

We assume stationarity (at least within a suitable time-window). Our working model is a generalization of the constant conditional correlation (CCC) GARCH model (Bollerslev, 1990),

$$\mathbf{X}_t = \Sigma_t \mathbf{Z}_t, \tag{4.1}$$

where we assume the following:

(A1) (innovations) $\{\mathbf{Z}_t\}_{t\in\mathbb{Z}}$ is a sequence of i.i.d. multivariate innovations with spherical distribution (e.g. multivariate normal) having mean zero and covariance matrix $\mathrm{Cov}(\mathbf{Z}_t) = I_d$. Moreover, $\mathbf{Z}_t$ is independent from $\mathcal{F}_{t-1} = \{\mathbf{X}_s; s \le t-1\}$.

(A2) (CCC construction) The conditional covariance matrix $V_t = \mathrm{Cov}(\mathbf{X}_t|\mathcal{F}_{t-1}) = \Sigma_t\Sigma_t^T$ is almost surely positive definite for all t. The typical element of V_t is $v_{t,ij} = \rho_{ij}(v_{t,ii}v_{t,jj})^{1/2}$ $(i, j = 1, \ldots, d)$. The parameter $\rho_{ij} = \mathrm{Corr}(X_{t,i}, X_{t,j}|\mathcal{F}_{t-1})$ equals the constant conditional correlation and hence $-1 \le \rho_{ij} \le 1$, $\rho_{ii} = 1$.

(A3) (functional form) The conditional variances are of the form

$$v_{t,ii} = \sigma_{t,i}^2 = \mathrm{Var}(X_{t,i}|\mathcal{F}_{t-1}) = F_i(\{X_{t-j,k};\ j = 1, 2, \ldots, k = 1, \ldots, d\})$$

where F_i takes values in $\mathbb{R}^+$.

Note that (A2) can be represented in matrix form as

$$\begin{aligned} &V_t = \Sigma_t\Sigma_t^T = D_t R D_t,\\ &D_t = \mathrm{diag}(\sigma_{t,1}, \ldots, \sigma_{t,d}),\ R = [\rho_{ij}]_{i,j=1}^d. \end{aligned}$$

For estimating the functions $F_i(\cdot)$ in (A3), we propose FGD and restrict $F_i(\cdot) : \mathbb{R}^{pd} \to \mathbb{R}^+$ with p finite, i.e. involving the first p lagged multivariate observations. Estimation of the correlations can be easily done via empirical moments of residuals.

To proceed with a FGD technique, we first specify a suitable loss function. Assuming multivariate normality of the innovations $\mathbf{Z}_t$, the negative log-likelihood (conditional on the first p variables) is

$$
\begin{aligned}
& - \sum_{t=p+1}^{n} \log\left((2\pi)^{-d/2} \det(V_t)^{-1/2} \exp(-\mathbf{X}_t^T V_t^{-1} \mathbf{X}_t/2)\right) \\
= & \sum_{t=p+1}^{n} \left(\log(\det(D_t)) + \frac{1}{2}(D_t^{-1}\mathbf{X}_t)^T R^{-1} (D_t^{-1}\mathbf{X}_t)\right) + n' d \log(2\pi)/2 + n' \log(\det(R))/2
\end{aligned}
$$

where D_t is diagonal with elements $\sqrt{F_i(\mathbf{X}_{t-p}^{t-1})}$ and $n' = n - p$. This motivates the following loss function

$$
\begin{aligned}
& \lambda_R(\mathbf{Y}, \mathbf{f}) = \log(\det(D(\mathbf{f})) + \frac{1}{2}(D(\mathbf{f})^{-1}\mathbf{Y})^T R^{-1} (D(\mathbf{f})^{-1}\mathbf{Y}) + \frac{1}{2}\log(\det(R)) + \frac{d}{2}\log(2\pi), \\
& D(\mathbf{f}) = \mathrm{diag}(f_1, \ldots, f_d). \qquad (4.2)
\end{aligned}
$$

(The terms $d\log(2\pi)/2$ and $\log(\det(R))/2$ are constants and could be dropped). As pointed out with the subscript, the loss function depends on the unknown correlation matrix R. Our FGD algorithm will be constructed iteratively by estimating R and using the loss function with the estimated R to get an estimate for all F_i's.

Having a (previous) estimate $\hat{\mathbf{F}} = (\hat{F}_1, \ldots, \hat{F}_d)$, we construct the following estimate for the correlation matrix R. Build the residuals

$$\hat{\varepsilon}_{t,i} = X_{t,i}/\hat{F}_i(\mathbf{X}_{t-1}, \ldots)^{1/2},\ t = p+1, \ldots, n$$

and define

$$\hat{R} = (n-p)^{-1} \sum_{t=p+1}^{n} \hat{\varepsilon}_t \hat{\varepsilon}_t^T,\ \hat{\varepsilon}_t = (\hat{\varepsilon}_{t,1}, \ldots, \hat{\varepsilon}_{t,d})^T. \qquad (4.3)$$

The partial derivatives of the loss function are

$$\frac{\partial \lambda_R(\mathbf{Y}, \mathbf{f})}{\partial f_i} = (f_i - \sum_{j=1}^{d} \frac{\gamma_{ij} y_i y_j}{f_i^{3/2} f_j^{1/2}})/2,\ i = 1, \ldots, d, \qquad (4.4)$$

where $[\gamma_{ij}]_{i,j=1}^d = R^{-1}$. This will be used when computing negative gradients (see Step 2 in the generic FGD algorithm) for every component $i = 1, \ldots, d$.

As a starting function, we propose to use the fit from a CCC-GARCH(1,1) model (Bollerslev, 1990) which is of the form (4.1) with (A3) specified to

$$F_i(X_{t-1}, X_{t-2}, \ldots) = \sigma_{t,i}^2 = \alpha_{0,i} + \alpha_{1,i} X_{t-1,i}^2 + \beta_{0,i} \sigma_{t-1,i}^2,\ i = 1, \ldots, d. \qquad (4.5)$$

For d large, the estimates are constructed with maximum likelihood from the d individual series. This ignores the more general correlation structure in R, causing some statistical decrease in efficiency, but gaining the advantage that the individual estimates are computable (in parallel) in very high dimensions d.

The FGD algorithm for multivariate volatility looks as follows.

FGD for multivariate volatility

Step 1 (initialization). Choose the starting function $\hat{F}_{i,0}(\cdot)$ and denote by $\hat{F}_{i,0}(t) = \hat{F}_{i,0}(\mathbf{X}_{t-1}, \mathbf{X}_{t-2}, \ldots)$ $(i = 1, \ldots, d)$. Compute $\hat{R}_0$ as in (4.3) using $\hat{\mathbf{F}}_0$. Set $m = 1$.

For every component $i = 1, \ldots, d$, do the following.

Step 2_i *(projection of component gradients to base learner).* Compute the negative gradient

$$U_{t,i} = -\frac{\partial \lambda_{\hat{R}_{m-1}}(\mathbf{X}_t, \mathbf{F})}{\partial F_i}|_{\mathbf{F}=\hat{\mathbf{F}}_{m-1}(t)},\ t = p+1, \ldots, n.$$

This is explicitly given in (4.4). Then, fit the negative gradient vector $U_i = (U_{p+1,i}, \ldots, U_{n,i})^T$ with a base learner, using always the first p time-lagged predictor variables (i.e. $\mathbf{X}_{t-p}^{t-1}$ is the predictor for $U_{t,i}$)

$$\hat{f}_{m,i}(\cdot) = \mathcal{S}_X(U_i)(\cdot).$$

Step 3_i *(line search).* Perform one-dimensional optimization for the step-length,

$$\hat{w}_{m,i} = \text{argmin} \sum_{t=p+1}^{n} \lambda_{\hat{R}_{m-1}}(\mathbf{X}_t, \hat{\mathbf{F}}_{m-1}(t) + w\hat{f}_{m,i}(\mathbf{X}_{t-p}^{t-1})).$$

($\hat{\mathbf{F}}_{m-1}(t) + w\hat{f}_{m,i}(\cdot)$ is defined as the function which is constructed by adding in the ith component only). This can be expressed more explicitly by using (4.2).

Step 4 (up-date). Select the best component as

$$i_m^* = \text{argmin}_i \sum_{t=p+1}^{n} \lambda_{\hat{R}_{m-1}}(\mathbf{X}_t, \hat{\mathbf{F}}_{m-1}(t) + \hat{w}_{m,i}\hat{f}_{m,i}(\mathbf{X}_{t-p}^{t-1})).$$

Up-date

$$\hat{\mathbf{F}}_m(\cdot) = \hat{\mathbf{F}}_{m-1}(\cdot) + \hat{w}_{m,i_m^*}\hat{f}_{m,i_m^*}(\cdot).$$

Then, compute the new estimate $\hat{R}_m$ according to (4.3) using $\hat{\mathbf{F}}_m$.

Step 5 (iteration). Increase m by one and iterate Steps 2–4 until stopping with $m = M$. This produces the FGD estimate

$$\hat{\mathbf{F}}_M(\cdot) = \hat{\mathbf{F}}_0(\cdot) + \sum_{m=1}^{M} \hat{w}_{m,i_m^*}\hat{f}_{m,i_m^*}(\cdot).$$

As in the generic algorithm, the stopping value M is chosen to optimize a measure for out-sample prediction.

Note that shrinkage as in (2.2) is often useful in Steps 2–4. As in the univariate case, the starting function $\hat{F}_0(\cdot)$ matters a lot for obtaining good volatility estimates.

A crucial difference to the multivariate (multi-class) FGD algorithm from Friedman et al. (2000), who propose to cycle through the dimensions in a systematic way one after the other, is that our construction is with candidate components in Steps 2_i and 3_i and choosing the component i^* in Step 4 which brings the most substantial improvement ("the steepest direction") in a single FGD iteration. Cycling through in a systematic way forces to add complexity of the FGD estimate for every component: but this isn't realistic if one time series is "of simpler structure" than others. We illustrate this for one realization of a 3-dimensional model as in (4.1): the volatilities are given by F_1 and F_2 from (4.8) below with fixed parameters $\alpha_1 = 0.1, \alpha_2 = 0.5, \alpha_3 = 0.2, \alpha_4 = 0.75, \alpha_5 = 0.5$ and F_3 from (4.9) below with fixed parameters $\alpha_1 = 0.1,\ \alpha_2 = 0.9,\ \alpha_3 = -1.5, \alpha_4 = 0.5$. Table

Model	Performance measure						
	OS $-$log-lik.	IS-L$_2$			OS-L$_2$		
		series$_1$	series$_2$	series$_3$	series$_1$	series$_2$	series$_3$
CCC-GARCH(1,1)	3269.1	207.50	206.89	197.71	147.71	173.78	203.81
FGD with trees using systematic cycling	3047.6	151.20	163.16	238.45	88.041	117.94	259.79
FGD with trees selecting best component i^*	3044.4	150.49	162.65	188.39	88.009	117.77	197.07

Table 4.1: Goodness of fit measures for one three-dimensional realization of size $n = 1000$ from model (4.1) with $\mathbf{Z}_t \sim \mathcal{N}_3(0, I)$ and individual conditional variances defined by (4.8) and (4.9). Out-sample negative log-likelihood as in (4.6) and individual OS-L$_2$ as in (3.6) are evaluated with test-set of size $n_{out} = 1000$.

4.1 impressively demonstrates that choosing the best component i^* brings substantial improvements for the third series. The goodness of fit criteria used are here the following. The out-sample negative log-likelihood function (the out-sample loss λ) is

$$\sum_{t=n+1}^{n+n_{out}} \lambda_{\hat{R}}(\mathbf{X}_t, \hat{\mathbf{F}}(t)), \tag{4.6}$$

where the estimates, denoted by a "$\widehat{\ }$" are based on the training data $\mathbf{X}_1, \ldots, \mathbf{X}_n$. The other criteria are the univariate in- and out-sample L_2-losses from (3.5) and (3.6).

4.1 Base learners with variable selection

Regarding the base learner $\mathcal{S}$, we have considered regression trees (Breiman at al., 1984). Fitting regression trees for the components in our multivariate FGD (Step 2_i) can be modified as in the univariate setting described in the Appendix. In very high dimensions, it is essential to use a base learner which selects only a few variables from a huge predictor space. Decision trees have this property: when having L terminal nodes, the decision tree base learner selects at most $L - 1$ different explanatory variables. In combination with FGD, we would then add to the starting function $\hat{F}_0$ an additive correction involving at most $M(L-1)$ different variables which may be much lower than dp which is the number of the p lagged predictor variables in every of the d time series.

4.2 Computational cost and parallelization

The computational complexity of our multivariate FGD algorithm (without initialization) is

$$Md \cdot \text{complexity}(\mathcal{S}) + Md \cdot \text{complexity}(\text{line search}) + M \cdot \text{complexity}(\hat{R})$$

The complexity of the estimate $\hat{R}$ in (4.3) is quadratic in the dimension but the computational cost of this moment estimator is not substantial relative to the other tasks. The numerical line search has to be done Md times which can contribute substantially to computing time when d is in the hundreds, M up to 100 (which is most often big enough) and n around 1000 (which is at the upper range, due to possible non-stationarity, when using daily financial data). Likewise, the base learner has to be fitted Md times. When d gets large, the increase of the factor Md is compounded by the fact that fitting the base learner becomes more costly. For example with decision trees, the computational complexity grows linearly in d. For d in the range of 100, M up to 100 and n about 1000, the total complexity is still feasible when using decision trees.

When d is in the order of 1000, the simple implementation of the FGD algorithm becomes quickly computationally expensive. Fortunately, it is very easy to implement a parallel version. Searching the best component i_m^* in Steps 2_i and 3_i requires consideration of all components $i = 1, \ldots, d$: this can be parallelized immediately, reducing the computational cost a lot. Furthermore, when using trees as base learner in high dimensions, their fitting, which requires visiting all components $i = 1, \ldots, d$ could also be substantially parallelized. While this is a bit more sophisticated, parallelization of the Steps 2_i and 3_i is immediate. Such relatively simple parallelizations make our FGD algorithm feasible in dimensions d in the thousands.

4.3 Numerical results

4.3.1 100-dimensional simulated data

We simulate a 100-dimensional series of sample size $n = 1000$ from model (4.1) with $\mathbf{Z}_t \sim \mathcal{N}_{100}(0, I)$ and various volatility functions F_j. One such function is the classical GARCH(1,1) volatility

$$\begin{aligned}
&\sigma_{t,i}^2 = F_i(X_{t-1,i}, \sigma_{t-1,i}^2) \text{ where} \\
&F_i(x, \sigma^2) = \alpha_0 + \alpha_1 x^2 + \beta\sigma^2, \text{ where} \\
&\alpha_0 \sim \text{Unif}([0, 0.2]),\ \alpha_1 \sim \text{Unif}([0.05, 0.15]),\ \beta \sim \text{Unif}([0.8, 0.84])
\end{aligned} \tag{4.7}$$

and $\alpha_0, \alpha_1, \beta$ mutually independent. Another function is from a threshold model

$$\begin{aligned}
&\sigma_{t,i}^2 = F_i(X_{t-1,i}, \sigma_{t-1,i}^2) \text{ where} \\
&F_i(x, \sigma^2) = \begin{cases} \alpha_1 + \alpha_2 x^2\ , & \text{if } x \le d_1 = 0, \\ 0.2 + \alpha_3 x^2 + \alpha_4 \sigma^2\ , & \text{if } x > d_1 = 0 \text{ and } \sigma^2 \le d_2 = 0.5, \\ 0.8 + \alpha_5 \sigma^2\ , & \text{if } x > d_1 = 0 \text{ and } \sigma^2 > d_2 = 0.5, \end{cases} \quad \text{where} \\
&\alpha_1 \sim \text{Unif}([0, 0.3]),\ \alpha_2 \sim \text{Unif}([0.4, 0.6]),\ \alpha_3 \sim \text{Unif}([0.1, 0.3]), \\
&\alpha_4 \sim \text{Unif}([0.6, 0.8]),\ \alpha_5 \sim \text{Unif}([0.4, 0.6])
\end{aligned} \tag{4.8}$$

($\alpha_1, \ldots, \alpha_5$ mutually independent). A third and a fourth function, in which we also allow for one cross-terms, are

$$\begin{aligned}
&\sigma_{t,i}^2 = F_i(X_{t-1,i}, X_{t-1,j}, \sigma_{t-1,i}^2) \text{ where} \\
&F_i(x, y, \sigma^2) = (\alpha_1 + 0.2\,|y| + \alpha_2 x^2) \cdot (0.8 \exp(\alpha_3\,|x|\,|\sigma|)) + (0.4x^2 + \alpha_4 \sigma^2)^{3/4}, \\
&\alpha_1 \sim \text{Unif}([0.05, 0.15]),\ \alpha_2 \sim \text{Unif}([0.8, 0.95]), \\
&\alpha_3 \sim \text{Unif}([-1.6, -1.4]),\ \alpha_4 \sim \text{Unif}([0.4, 0.6])
\end{aligned} \tag{4.9}$$

($\alpha_1, \ldots, \alpha_4$ mutually independent), and

$$\begin{aligned}
&\sigma_{t,i}^2 = F_i(X_{t-1,i}, X_{t-1,j}, \sigma_{t-1,i}^2) \text{ where} \\
&F_i(x, y, \sigma^2) = (0.1 + \alpha_1\,|y|^3) \cdot \exp(\alpha_2 x^2) + \alpha_3 (\sigma^2)^{3/4}, \\
&\alpha_1 \sim \text{Unif}([0.1, 0.2]),\ \alpha_2 \sim \text{Unif}([-0.1, 0]),\ \alpha_3 \sim \text{Unif}([0.8, 0.9])
\end{aligned} \tag{4.10}$$

($\alpha_1, \alpha_2, \alpha_3$ mutually independent), where the component $j \in \{1, \ldots, d\} \setminus i$ is chosen randomly. Each of the volatility functions F_i is randomly chosen with probability 1/4, independent of each other. Note that also the coefficients in these functions are randomly chosen. The constant conditional correlation matrix R is chosen to mimic the one of real log-returns. This model is "fairly close" to a CCC-GARCH(1,1) model since half of the volatility functions involve only auto-dependence (no dependence on a cross-series), a quarter of them actually being linear GARCH-type, and the other half involve only one other cross-series.

The results are displayed in Table 4.2 and Figure 4.1 (test set size is $n_{out} = 1000$).

Model	Performance measure			
	OS $-$log-lik.	ave(OS-$\mathrm{L}_2^{(j)}$)	max gain in OS-L_2	min gain in OS-L_2
GARCH(1,1)	121636.8	447.7096	–	–
FGD with tree using $p = 2$, $\nu = 0.5,\ L = 5$	119763.0 (1.5%)	410.3455 (8.3%)	27.2%	-12.1%

Table 4.2: Goodness of fit measures for one 100-dimensional realization of size $n = 1000$ as described above. The measures are defined as in (4.6), as the average over all $d = 100$ individual series as in (3.6) and the extremal relative individual gains with respect to (3.6); test-set size is $n_{out} = 1000$ and relative gains with FGD are given in parentheses.

We observe only a small gain of FGD with trees over the CCC-GARCH(1,1) prediction with respect to the negative log-likelihood, because the signal to noise level is low, but a more substantial gain when looking at individual outsample L_2-losses as in (3.6) where the observation noise is not present. Of course, we cannot expect to learn in all $d = 100$ components with sample size $n = 1000$ (or here in all 75 components which are not of linear GARCH(1,1)-type). On average, the OS-L_2 gain over all $d = 100$ components is 8.3%: this gain will generally decrease when dimension d increases and keeping sample size n fixed. However, we also see from Figure 4.1, that FGD mainly improves (in absolute terms) at those components where the CCC-GARCH(1,1) predictions are poorest. This

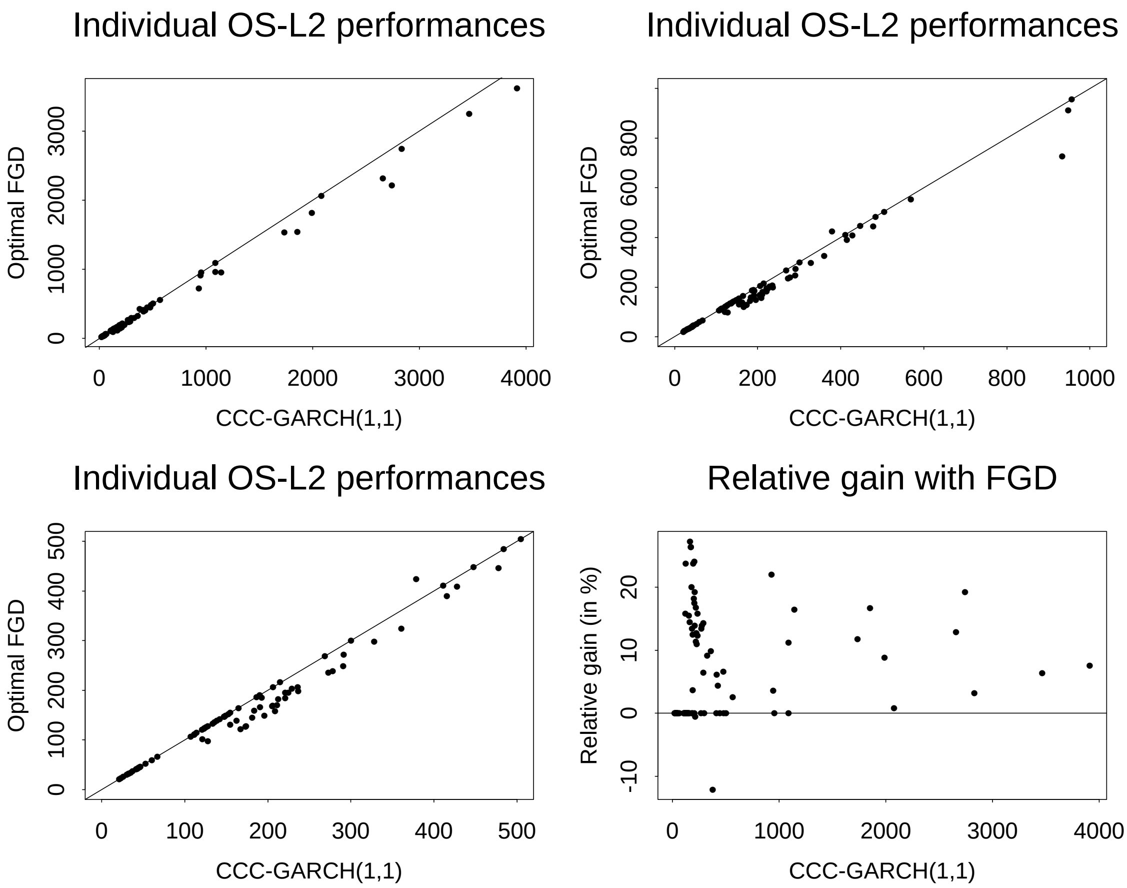

Figure 4.1: Individual OS-L_2 measures as in (3.6) for one 100-dimensional realization of size $n = 1000$ as described above. First three panels: performance of CCC-GARCH(1,1) on x-axis and of FGD with trees on y-axis with the line indicating equal performance, on different scales. Lower right panel: relative gains with FGD with trees of individual OS-L_2 measures (see 3.6) on y axis versus performance of CCC-GARCH(1,1) on x-axis.

is consistent with the intuition that functional gradient descent improves the "hardest" cases. Regarding the relative gains with FGD: they range from -12.1 to 27.2% and quite many large relative gains are realized where the CCC-GARCH predictions are fairly low (see lower right panel of Figure 4.1).

4.3.2 A seven-dimensional real data example

We consider here real daily return data from seven financial indices. The dimensionality $d = 7$ is "mid-range" allowing for a representation of the results which is easy to survey. FGD will also turn out to be useful in such orders of dimensions. The data comprises 1500 daily returns from the US DJIA, the French CAC40, the German DAX, the Italian BCI, the Dutch CBS, the British FTAS and the Japanese NIKKEI index, during the period January 31, 1990 until September 9, 1996: we use the first 1000 time points for training (fitting) and the remaining 500 for out-sample testing (evaluation).

We consider two goodness of fit measures for such real data. One is the negative out-sample log-likelihood in (4.6). Alternatively, we also look at the individual out-sample prediction L_2 losses,

$$\text{OS-PL}_2 = \sum_{t=1001}^{1500} |X_{t,i}^2 - \hat{F}_i(t)|^2 \ (i = 1, \ldots, d).$$

The IS-PL$_2$ is defined analogously using the training sample only. The results are given in Table 4.3.

	Performance measure					
	CCC-GARCH(1,1)			FGD with trees		
	OS $-$log-lik.	IS-PL$_2$	OS-PL$_2$	OS $-$log-lik.	IS-PL$_2$	OS-PL$_2$
global	5614.13			5547.46		
DJIA		1082.89	5874.58		1069.32	5852.40
CAC40		11191.4	21915.1		11133.9	21910.8
DAX		10382.6	41318.4		10261.8	41025.7
BCI		8753.99	36145.6		8274.50	36319.9
CBS		5725.22	25191.9		5725.22	25191.9
FTAS		2109.78	5434.23		2060.71	5391.61
NIKKEI		45610.1	25137.2		45081.0	24936.1

Table 4.3: Goodness of fit measures for a seven-dimensional real data example. The optimal parameters in FGD with trees are $p = 3$ lagged values, shrinkage $\nu = 0.5$ and $L = 3$ terminal nodes.

With real data, differences in the out-sample negative log-likelihood or OS-PL$_2$ can be small between different methods; see also at the end of section 3.1. The out-sample criterion is here generally denoted by

$$\sum_{t=1}^{n_{out}} L_t,$$

where L_t is the out-sample loss for out-sample prediction at time t depending on a technique or model, i.e. $L_t = L_{t,\text{model}}$ (we will only consider $L_t = \lambda_{\hat{R}}(\mathbf{X}_t, \hat{\mathbf{F}}(t))$). The differences between two models with respect to out-sample performance is then

$$\Delta(\text{model}_1, \text{model}_2) = \sum_{t=1}^{n_{out}} D_t, \ D_t = L_{t,\text{model}_1} - L_{t,\text{model}_2}.$$

We aim to test whether $\mathbb{E}[D_t] = 0$ against one- (or two-) sided alternatives. We consider versions of the t- and sign-test, adapted to the case of dependent observations D_t.

t-type test	sign-type test
-2.77 (0.003)	-2.76 (0.003)

Table 4.4: Test-statistics and P-values (in parentheses) for one-sided testing for differences in out-sample negative log-likelihood of the FGD with trees and the CCC-GARCH predictions. Negative test-statistics favor the FGD method.

The t-type test statistic is

$$n_{out}^{1/2}\frac{\overline{D}}{\hat{\sigma}_{D,\infty}},\ \overline{D} = n_{out}^{-1}\sum_{t=1}^{T} D_t,$$
$$\hat{\sigma}^2_{D,\infty} = 2\pi\hat{f}_D(0) \tag{4.11}$$

where $\hat{f}_D(0)$ is an estimate of $f_D(0) = 2\pi\sum_k |cov(D_0, D_k)$, the spectral density at frequency zero of the process $\{D_t\}_t$, where we assume that $\{D_t\}_t$ is stationary and suitably regular, so that

$$n_{out}^{1/2}(\overline{D} - \mathbb{E}[D_t]) \Rightarrow \mathcal{N}(0, \sigma^2_{D,\infty})$$

From this, we obtain asymptotically the standard normal distribution of the test-statistic in (4.11) under the null-hypothesis and hence the ingredients to perform the test.

The sign-type test statistic is

$$n_{out}^{1/2}\frac{\overline{W} - 1/2}{\hat{\sigma}_{W,\infty}},\ \overline{W} = n_{out}^{-1}\sum_{t=1}^{T} \mathbb{I}_{[D_t>0]},$$
$$\hat{\sigma}^2_{D,\infty} = 2\pi\hat{f}_D(0) \tag{4.12}$$

where $\hat{f}_W(0)$ is an estimate of the spectral density at frequency zero of the process $\{\mathbb{I}_{[D_t>0]}\}_t$, where we assume that $\{\mathbb{I}_{[D_t>0]}\}_t$ is stationary and suitably regular, so that the central limit theorem holds. This then implies that the test-statistic in (4.12) is asymptotically standard normal under the null-hypothesis $H_0 : p = \mathbb{P}[D_t > 0] = 1/2$ which allows to perform the test. For both tests, estimation of the spectral density at zero is done via smoothing the periodogram.

We apply both tests for comparing the FGD with trees and the CCC-GARCH predictions. The alternative is one-sided where FGD with trees has lower out-sample loss. The results are given in Table 4.4. Thus, the small difference of 1.2% in the out-sample log-likelihoods of FGD and CCC-GARCH(1,1) from Table 4.3 turns out to be highly significant.

5 Conclusions

We have presented an FGD algorithm which is a technique for estimation of the conditional covariance matrix in (1.1). It is computationally feasible in multivariate problems with several hundreds up to thousands of return series. The algorithm is constructed from a generic algorithm: hence, other FGD algorithms can be derived aiming to learn other, typically very high-dimensional, problems.

We have demonstrated on some data-sets (synthetic and real) that our FGD algorithm significantly outperforms the predictions from the CCC-GARCH(1,1) model (Bollerslev, 1990). The latter model has generated the starting functions in our FGD and hence, it is not so surprising that we could observe improvements.

This seems generally the attractive feature of FGD. We choose a reasonable model for generating (estimating) the starting functions in FGD, and then we try – often successfully – to improve the initial basis model with a couple FGD iterations. According to the heuristics of a steepest functional gradient, the improvements with FGD are mainly expected at those components where the initial basis model performs poorly, as we demonstrated on a 100-dimensional data-set. Therefore, FGD is not necessarily restricted to the framework of constant conditional correlation (CCC).

Appendix

When using regression tree base learners in FGD, we can take advantage of the fact that the learner is a partitioning method which allows a more (statistically) efficient implementation for fitting with respect to the best improvement of the loss function λ.

FGD with tree learners

(Formulated with response variables Y_i and explanatory variables X_i, see section 3).

Modified steps 2 and 3. Given a negative gradient vector U, fit a regression tree to U by least squares. This produces a partition $\{\mathcal{R}_1, \dots, \mathcal{R}_k\}$ of the predictor space $\mathbb{R}^p$

$$\cup_{j=1}^k \mathcal{R}_j = \mathbb{R}^p,\ \mathcal{R}_i \cap \mathcal{R}_j = \emptyset\ (i \neq j).$$

(I.e., the partition is such that $\sum_{i=1}^n (U_i - \sum_{j=1}^k \hat{\beta}_j \mathbb{I}_{[X_i \in \mathcal{R}_j]})^2$ is minimal). Then, proceed with line searches for all k partition cells,

$$\hat{\gamma}_{m,j} = \mathrm{argmin}_\gamma \sum_{i; X_i \in \mathcal{R}_j} \lambda(Y_i, \hat{F}_{m-1}(X_i) + \gamma \mathbb{I}_{[X_i \in \mathcal{R}_j]}),\ j = 1, \dots, k.$$

Finally, up-date

$$\hat{F}_m(x) = \hat{F}_{m-1}(x) + \sum_{j=1}^k \hat{\gamma}_{m,j} \mathbb{I}_{[x \in \mathcal{R}_j]}.$$

This is the same generic tree algorithm as in Friedman (2001).

References

[1] Aguilar, O. and West, M. (2000). Bayesian dynamic factor models and variance matrix discounting for portfolio allocation. J. of Business and Economic Statistics **18**, 338–357.

[2] Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. J. of Econometrics **31**, 307–327.

[3] Bollerslev, T. (1990). Modelling the coherence in short-run nominal exchange rates: a multivariate generalized ARCH model. The Review of Economics and Statistics **72**, 498–505.

[4] Breiman, L. (1999). Prediction games & arcing algorithms. Neural Computation **11**, 1493-1517.

[5] Breiman, L., Friedman, J.H., Olshen, R.A. & Stone, C.J. (1984). *Classification and Regression Trees*. Wadsworth, Belmont (CA).

[6] Bühlmann, P. and Yu, B. (2001). Boosting with the L_2-loss: regression and classification. Preprint, ETH Zürich.

[7] Chib, S., Nardari, F. and Shephard, N. (1999). Analysis of high dimensional multivariate stochastic volatility models. Preprint, University of Oxford.

[8] Engle, R.F. and Kroner, K.F. (1995). Multivariate simultaneous generalized ARCH. Econometric Theory **11**, 122–150.

[9] Engle, R.F., Ng, V.K. and Rothschild, M. (1990). Asset pricing with a factor ARCH covariance structure: empirical estimates for treasury bills. J. Econometrics **45**, 231–238.

[10] Freund, Y. and Schapire, R.E. (1996). Experiments with a new boosting algorithm. In Machine Learning: Proc. Thirteenth International Conference, pp. 148–156. Morgan Kauffman, San Francisco.

[11] Friedman, J.H. (2001). Greedy function approximation: a gradient boosting machine. To appear in the Annals of Statistics.

[12] Friedman, J.H., Hastie, T. and Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. Annals of Statistics **28**, 337–407 (with discussion).

[13] Friedman, J.H. and Stuetzle, W. (1981). Projection pursuit regression. J. American Statistical Association **76**, 817–823.

[14] Harvey, A.C., Ruiz, E. and Shephard, N. (1994). Multivariate stochastic variance models. Rev. Economic Studies **61**, 247–264.

[15] Lin, W.-L. (1992). Alternative estimators for factor GARCH models – a Monte Carlo comparison. J. Applied Econometrics **7**, 259–279.

[16] Mason, L., Baxter, J. Bartlett, P. and Frean, M. (1999). Functional gradient techniques for combining hypotheses. In *Advances in Large Margin Classifiers*. MIT Press.