2

Robust Regression with Optimisation Heuristics

Manfred Gilli and Enrico Schumann

Department of Econometrics, University of Geneva,
Bd du Pont d'Arve 40, 1211 Geneva 4, Switzerland
Manfred.Gilli@unige.ch, Enrico.Schumann@unige.ch

Summary. Linear regression is widely-used in finance. While the standard method to obtain parameter estimates, Least Squares, has very appealing theoretical and numerical properties, obtained estimates are often unstable in the presence of extreme observations which are rather common in financial time series. One approach to deal with such extreme observations is the application of robust or resistant estimators, like Least Quantile of Squares estimators. Unfortunately, for many such alternative approaches, the estimation is much more difficult than in the Least Squares case, as the objective function is not convex and often has many local optima. We apply different heuristic methods like Differential Evolution, Particle Swarm and Threshold Accepting to obtain parameter estimates. Particular emphasis is put on the convergence properties of these techniques for fixed computational resources, and the techniques' sensitivity for different parameter settings.

2.1 Introduction

Linear regression is a widely-used tool in finance. A common practice is, for instance, to model the returns of single assets as a linear combination of the returns of various types of 'factors'. Such regressions can then be used to explain past returns, or in attempts to forecast future returns. In financial economics, such factor models are the main tools for asset pricing, for instance in the Capital Asset Pricing Model (CAPM), or in the Arbitrage Pricing Theory (APT). Even if these models, when interpreted as equilibrium models, do not hold in practice, the underlying regressions are still valuable. A main area of application is risk management, where the regression estimates can be used to construct variance–covariance matrices. There is considerable evidence of the usefulness of such models in this context [6].

Regression models may not only be used to inform financial decisions by analysing assets, but may be more explicitly used when constructing portfolios. For instance, a possible approach to replicate a portfolio or an index is to find investable assets whose returns 'explain' the chosen regressand (eg, the index); see for instance [26]. Assume we have p assets, and let the symbol x_i stand for the return of asset i at some point in time; we use x_i^* for the excess return over a constant riskfree rate. If a riskfree asset exists, mean–variance portfolio optimisation reduces to finding the portfolio with the maximum Sharpe ratio. This optimisation problem can be rewritten as

$$1 = \theta_1 x_1^* + \theta_2 x_2^* + \cdots + \theta_p x_p^* + \epsilon$$

A. Brabazon et al. (Eds.): Natural Computing in Computational Finance: Vol. 3, SCI 293, pp. 9–30.
springerlink.com

where θ_i are the coefficients to be estimated, and ϵ holds the errors. Estimating the θ_i with Least Squares and rescaling them to conform with the budget constraint is equivalent to solving a mean–variance problem for the tangency portfolio weights, see [4]. The approach is outlined in the Appendix.

We can also find the global minimum-variance portfolio by running a regression [19]. We write the portfolio return as the sum of its expectation μ and an error ϵ, hence

$$\mu + \epsilon = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_p x_p \,.$$

Imposing the budget constraint $\sum \theta = 1$ and rearranging we get

$$x_p = \mu + \theta_1 (x_p - x_1) + \theta_2 (x_p - x_2) + \cdots + \theta_{p-1} (x_p - x_{p-1}) + \epsilon \,.$$

We can directly read off the portfolio weights from the regression; the weight of the pth position is determined via the budget constraint.

Finally, linear models are used to evaluate the ex post performance of investment managers: since [28], 'style analysis' has become a building block in performance measurement and evaluation. The regression coefficients are then interpreted as portfolio weights and the residuals as managerial skill (or luck).

The standard method to obtain parameter estimates for a linear regression model is Least Squares (LS). LS has very appealing theoretical and numerical properties, but the resulting estimates are often unstable if there exist extreme observations which are common in financial time series [5, 21, 11]. In fact, a few or even a single extreme data point can heavily influence the resulting estimates. A much-studied example is the estimation of β-coefficients for the CAPM, where small changes in the data (resulting, for instance, from a moving-window scheme) often lead to large changes in the estimated β-values. Earlier contributions in the finance literature suggested some form of shrinkage of extreme coefficients towards more reasonable levels, with different theoretical justifications (see for example [2, 32, 20]). An alternative approach, which we will deal with in this Chapter, is the application of robust or resistant estimation methods [5, 22].

There is of course a conceptual question as to what constitutes an extreme observation or outlier in financial time series. Extreme returns may occur rather regularly, and completely disregarding such returns by dropping or winsorising them could mean to throw away information. Errors in the data, though, for example stock splits that have not been accounted for, are clearly outliers. Such data errors occur on a wide scale, even with commercial data providers [18]. Hence in particular if data are processed automatically, alternative techniques like robust estimation methods may be advisable.

In this Chapter, we will discuss the application of robust estimators. Such estimators were specially designed not to be influenced too heavily by outliers, even though this characteristic often comes at the price of low efficiency if the data actually contain no outliers. Robust estimators are often characterised by their breakdown value. In words, the breakdown point is the smallest percentage of contaminated (outlying) data that may cause the estimator to be affected by an arbitrary bias [25]. While LS has a breakdown point of 0%, other estimators have breakdown points of up to 50%. Unfortunately, the estimation becomes much more difficult, and for many models only approximative solutions exist. We will describe the application of heuristics to such optimisation problems. More precisely, we will compare different optimisation methods, namely Differential

Evolution, Particle Swarm, and Threshold Accepting. All three methods are general-purpose heuristics and have been successfully applied to a wide range of problems, see for instance [23], [33].

The remaining Chapter is structured as follows: Section 2.2 will introduce the linear regression model and several alternative optimisation criteria for parameter estimation. Section 2.3 will discuss numerical estimation strategies, ie, we will discuss different optimisation procedures. In Section 2.4 then, we use the Monte-Carlo setup from [27] to test the convergence behaviour of the different optimisation methods when used for a specific estimator, Least Median of Squares. Section 2.5 concludes.

2.2 The Linear Regression Model

We consider the linear regression model

$$y = \begin{bmatrix} x_1 & \cdots & x_p \end{bmatrix} \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_p \end{bmatrix} + \epsilon .$$

Here, y is a vector of n observations of the independent variable; there are p regressors whose observations are stored in the column vectors x_j. We will usually collect the regressors in a matrix $X = \begin{bmatrix} x_1 & \cdots & x_p \end{bmatrix}$, and write θ for the vector of all coefficients. The jth coefficient is denoted by θ_j. We will normally include a constant as a regressor, hence x_1 will be a vector of ones. The residuals r (ie, the estimates for the ϵ), are computed as

$$r = y - X\hat{\theta}$$

where $\hat{\theta}$ is an estimate for θ. Least Squares (LS) requires to minimise the sum or, equivalently, the mean of the squared residuals, hence the estimator is defined as

$$\hat{\theta}_{\text{LS}} = \underset{\theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} r_i^2 .$$

The advantage of this estimator is its computational tractability: the LS solution is found by solving the system of normal equations

$$(X'X)\theta = X'y$$

for θ.

Rousseeuw [24] suggested to replace the mean of the squared residuals with their median. The resulting Least Median of Squares (LMS) estimator can be shown to be less sensitive to outliers than LS; in fact, LMS's breakdown point is almost 50%. More formally, LMS is defined as

$$\hat{\theta}_{\text{LMS}} = \underset{\theta}{\operatorname{argmin}} \operatorname{median}(r^2) .$$

LMS can be generalised to the Least Quantile of Squares (LQS) estimator. Let Q_q be the qth quantile of the squared residuals, that is

$$Q_q = \mathrm{CDF}^{-1}(q) = \min\{r_i^2 \mid \mathrm{CDF}(r_i^2) \geq q\}, \tag{2.1}$$

where q may range from 0% to 100% (we drop the %-sign in subscripts). Hence the LMS estimator becomes

$$\hat{\theta}_{\mathrm{LMS}} = \underset{\theta}{\operatorname{argmin}}\, Q_{50}(r^2),$$

and more generally we have

$$\hat{\theta}_{\mathrm{LQS}} = \underset{\theta}{\operatorname{argmin}}\, Q_q(r^2).$$

For a given sample, several numbers satisfy definition (2.1), see [17]. A convenient approach is to work directly with the order statistics $[r^2_{[1]}\ r^2_{[2]}\ \ldots\ r^2_{[n]}]'$. For LMS, for instance, the maximum breakdown point is achieved not by minimising $Q_{50}(r^2)$, but by defining

$$h = \left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{p+1}{2} \right\rfloor \tag{2.2}$$

and minimising $r^2_{[h]}$ [24][p. 873].

The Least Trimmed Squares (LTS) estimator requires to minimise the order statistics of r^2 up to some maximum order k. Formally,

$$\hat{\theta}_{\mathrm{LTS}} = \underset{\theta}{\operatorname{argmin}} \frac{1}{k} \sum_{i=1}^{k} r^2_{[i]}.$$

To achieve a high breakdown value, the number k is set to roughly $\lfloor 1/2(n+p+1) \rfloor$, or the order statistic defined in Equation (2.2).

LQS and LTS estimators are sometimes called 'resistant' estimators, since they do not just reduce the weighting of outlying points, but essentially ignore them. This property in turn results in a low efficiency if there are no outliers. However, we can sometimes exploit this characteristic when we implement specific estimators.

2.3 Estimation

2.3.1 Strategies

Robust estimation is computationally more difficult than LS estimation. A straightforward estimation strategy is to directly map the coefficients of a model into the objective function values, and then to evolve the coefficients according to a given optimisation method until a 'good' solution is found. For LMS, for instance, we may start with a 'guess' of the parameters θ and then change θ iteratively until the median squared residual cannot be reduced any further. We will refer to this strategy as the 'direct approach'.

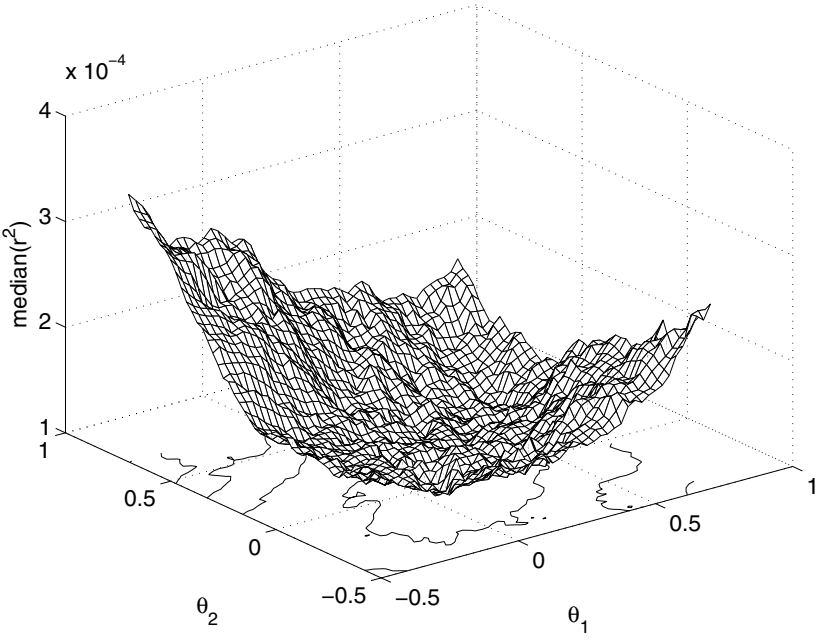


Fig. 2.1. Search space for LMS.

The difficulty with the direct approach arises from the many local minima that the objective function exhibits. This is illustrated in Figure 2.1 which shows the mapping from a given set of coefficients into the median squared residual (ie, the search space) for a CAPM regression $y = \theta_1 + \theta_2 x + \epsilon$ (here y is the excess return of a specific asset over the riskfree rate, and x the excess return of the market).

Heuristic methods deploy different strategies to overcome such local minima. We will compare three different techniques – Differential Evolution, Particle Swarm, and Threshold Accepting – for the direct approach.

Since many resistant estimators essentially fit models on only a subset of the data, we may also associate such subsets with particular objective function values – hence transform the estimation into a combinatorial problem. An intuitive example is the LTS estimator: since the objective is to minimise the sum of the k smallest squared residuals, we could also, for every subset of size k, estimate LS-coefficients. The subset with the minimum objective function will give us the exact solution to the problem. Since such a complete enumeration strategy is clearly infeasible for even moderately-sized models, we will investigate an alternative search strategy based on Threshold Accepting. We refer to this estimation strategy as the 'subset approach'.

In the remainder of this Chapter, we will limit ourselves to LMS estimation. The direct approach is, however, applicable to any estimation criterion that allows to directly connect the coefficients to the residual vector r. The subset approach presented later is applicable to LQS estimation; it could easily be modified (in fact, simplified) for LTS. Next we outline the different algorithms.

2.3.2 Differential Evolution

Differential Evolution (DE) was developed for continuous optimisation problems [29], we outline the procedure in Algorithm 2.1. DE evolves a population of n_P solutions, stored in real-valued vectors of length p (ie, the number of coefficients of the regression model). The population P may be visualised as a matrix of size $p \times n_P$, where each column holds one candidate solution. In every iteration (or 'generation'), the algorithm goes through the columns of this matrix and creates a new candidate solution for each existing solution $P^{(0)}_{\cdot,i}$. This candidate solution is constructed by taking the difference between two other solutions, weighting this difference by a parameter F, and adding it to a third solution. Then an element-wise crossover takes place with probability CR between this auxiliary solution $P^{(v)}_{\cdot,i}$ and the existing solution $P^{(0)}_{\cdot,i}$ (the symbol ζ represents a random variable that is uniformly distributed between zero and one). If this final candidate solution $P^{(u)}_{\cdot,i}$ is better than $P^{(0)}_{\cdot,i}$, it replaces it; if not, the old solution $P^{(0)}_{\cdot,i}$ is kept.

Algorithm 2.1. Differential Evolution.

initialise parameters n_P, n_G, F and CR;
initialise population $P^{(1)}_{j,i}$, $j = 1,\ldots,p$, $i = 1,\ldots,n_P$;
for $k = 1$ *to* n_G **do**
  $P^{(0)} = P^{(1)}$;
  for $i = 1$ *to* n_P **do**
    generate $\ell_1, \ell_2, \ell_3 \in \{1,\ldots,n_P\}$, $\ell_1 \neq \ell_2 \neq \ell_3 \neq i$;
    compute $P^{(v)}_{\cdot,i} = P^{(0)}_{\cdot,\ell_1} + \mathrm{F} \times (P^{(0)}_{\cdot,\ell_2} - P^{(0)}_{\cdot,\ell_3})$;
    for $j = 1$ *to* p **do**
      if $\zeta < \mathrm{CR}$ **then** $P^{(u)}_{j,i} = P^{(v)}_{j,i}$ **else** $P^{(u)}_{j,i} = P^{(0)}_{j,i}$;
    end
    if $\Phi(P^{(u)}_{\cdot,i}) < \Phi(P^{(0)}_{\cdot,i})$ **then** $P^{(1)}_{\cdot,i} = P^{(u)}_{\cdot,i}$ **else** $P^{(1)}_{\cdot,i} = P^{(0)}_{\cdot,i}$;
  end
end

2.3.3 Particle Swarm Optimisation

The narrative for Particle Swarm Optimisation (PS) is based on swarms of animals like birds or fish that look for food [9]. Like DE, PS is applicable to continuous problems; Algorithm 2.2 details the procedure. We have, again, a population that comprises n_P solutions, stored in real-valued vectors. In every generation, a solution is updated by adding another vector called velocity v_i. We may think of a solution as a position in the search space, and of velocity as a direction into which the solution is moved. Velocity changes over the course of the optimisation, the magnitude of change is the sum of two components: the direction towards the best solution found so far by the particular solution, $Pbest_i$, and the direction towards the best solution of the whole population, $Pbest_{\text{gbest}}$. These two directions are perturbed via multiplication with a uniform random variable ζ and constants $c_{(\cdot)}$, and summed, see Statement 2.2. The vector so obtained is added to the previous v_i, the resulting updated velocity is added to the respective

Algorithm 2.2. Particle Swarm.

initialise parameters n_{P}, n_{G}, δ, c_1 and c_2;
initialise particles $P_i^{(0)}$ and velocity $v_i^{(0)}$, $i = 1, \ldots, n_{\mathrm{P}}$;
evaluate objective function $F_i = \Phi(P_i^{(0)})$, $i = 1, \ldots, n_{\mathrm{P}}$;
$Pbest = P^{(0)}$, $Fbest = F$, $Gbest = \min_i(F_i)$, $gbest = \operatorname{argmin}_i(F_i)$;
for $k = 1$ *to* n_{G} **do**
  for $i = 1$ *to* n_{P} **do**
    $\Delta v_i = c_1 \times \zeta_1 \times (Pbest_i - P_i^{(k-1)}) + c_2 \times \zeta_2 \times (Pbest_{\mathrm{gbest}} - P_i^{(k-1)})$;
    $v_i^{(k)} = \delta v^{(k-1)} + \Delta v_i$;
    $P_i^{(k)} = P_i^{(k-1)} + v_i^{(k)}$;
  end
  evaluate objective function $F_i = \Phi(P_i^{(k)})$, $i = 1, \ldots, n_{\mathrm{P}}$;
  for $i = 1$ *to* n_{P} **do**
    if $F_i < Fbest_i$ **then** $Pbest_i = P_i^{(k)}$ and $Fbest_i = F_i$;
    if $F_i < Gbest$ **then** $Gbest = F_i$ and $gbest = i$;
  end
end

solution. In some implementations, the velocities are reduced in every generation by setting the parameter δ to a value smaller than unity.

2.3.4 Threshold Accepting (Direct Approach)

Threshold Accepting (TA) is a descendant of Simulated Annealing and was introduced by [8]. Other than DE and PS, TA is a so-called trajectory method and evolves only a single solution. It is based on a local search [14] but may, like Simulated Annealing, also move 'uphill' in the search space. More specifically, it accepts new solutions that are inferior when compared with the current solution, as long as the deterioration does not exceed a specified threshold, thus the method's name. Over time, this threshold decreases to zero, and so TA turns into a classical local search. Algorithm 2.3 describes the procedure; for an in-depth description see [33].

Algorithm 2.3. Threshold Accepting.

initialise n_{Rounds} and n_{Steps};
compute threshold sequence τ;
randomly generate current solution θ^{c};
for $r = 1 : n_{\mathrm{Rounds}}$ **do**
  for $i = 1 : n_{\mathrm{Steps}}$ **do**
    generate $\theta^{\mathrm{n}} \in \mathcal{N}(\theta^{\mathrm{c}})$ and compute $\Delta = \Phi(\theta^{\mathrm{n}}) - \Phi(\theta^{\mathrm{c}})$;
    if $\Delta < \tau_r$ **then** $\theta^{\mathrm{c}} = \theta^{\mathrm{n}}$;
  end
end
$\theta^{\mathrm{sol}} = \theta^{\mathrm{c}}$;

Here, θ^c denotes the current solution, and θ^n is the 'new' (or neighbour) solution. For each of the n_{Rounds} thresholds, stored in the vector τ, the algorithm performs n_{Steps} iterations, so the number of objective function evaluations is $n_{\text{Rounds}} \times n_{\text{Steps}}$.

Algorithm 2.4. Threshold Accepting – Neighbourhood definition.

$\theta^n = \theta^c$;
randomly select $j \in \{1, \ldots, p\}$;
randomly generate $\zeta \in [-z, z]$;
$\theta^n_j = \theta^c_j + \zeta \times (1 + |\theta^c_j|)$;

Neighbourhood Definition

While TA was originally introduced for combinatorial (ie, discrete) problems, it can easily be modified for continuous functions. We implement the neighbourhood function $\mathcal{N}$ as a small perturbation of the current coefficients vector. We use a random step size that is proportional to the respective coefficient (see Algorithm 2.4). Variations are possible; [35] for example suggest to shrink the step size over time.

The constant 1 is added in Statement 2.4 to make a sign-change for the given parameter more probable: without such a constant, when a coefficient gets closer to zero in absolute terms, its variation also goes to zero.

Threshold Sequence

To compute the threshold sequence we take a random walk through the solution space under the specified neighbourhood function and record the changes in the objective function. The thresholds are then equidistant quantiles of the distribution of the absolute values of the changes. For the rationale of this approach see [34, 15].

This procedure requires the number of thresholds n_{Rounds} to be set in advance. We set n_{Rounds} to 10, even though TA is robust for other choices. There is some evidence, though, that for very small numbers of thresholds, for instance 2 or 3, the performance of the algorithm deteriorates [13].

Algorithm 2.5. Computing the threshold sequence.

randomly choose θ^c;
for $i = 1 : n_{\text{Deltas}}$ **do**
  compute $\theta^n \in \mathcal{N}(\theta^c)$ and $\Delta_i = |\Phi(\theta^c) - \Phi(\theta^n)|$;
  $\theta^c = \theta^n$;
end
compute empirical distribution CDF of Δ_i, $i = 1, \ldots, n_{\text{Deltas}}$;
compute threshold sequence $\tau_r = \text{CDF}^{-1}\left(\frac{n_{\text{Rounds}} - r}{n_{\text{Rounds}}}\right)$, $r = 1, \ldots, n_{\text{Rounds}}$;

Algorithm 2.6. Chebyshev regression for $p+1$ subset.

solve $(X_s'X_s)\theta = X_s'y_s$ for θ;
compute $r_s = y_s - X_s\theta$;
compute $\omega = \sum r_s^2 / \sum |r_s|$;
compute $\sigma = \text{sign}(r_s)$;
compute $y_s^* = y_s - \omega\sigma$;
solve $(X_s'X_s)\theta = X_s'y_s^*$ for θ;
$\theta_c = \theta$;

2.3.5 Threshold Accepting (Subset Approach)

Let $r_{[h]}^2$ denote the median order statistic of the squared residuals. [30] noted that an estimator that minimises $r_{[h]}^2$ is equivalent to an estimator that minimises the largest squared residual for a subset of size h. This is almost equivalent to the so-called Chebyshev estimator for this subset, defined by

$$\hat{\theta}_c = \underset{\theta}{\text{argmin}} \max |r_i| .$$

(Only 'almost equivalent' because of using the absolute value instead of squaring the residuals.) A convenient fact about $\hat{\theta}_c$ is that there exists also a subset of size $p+1$ that yields the same fit as a h-subset. More generally, the LQS estimator for any order statistic h (not just LMS) corresponds to a Chebyshev estimate of some subset of size $p+1$. Thus a solution with this approach is identified by $p+1$ indices, pointing to specific rows in $[y\ X]$. Then, by computing $\hat{\theta}_c$ for this subset, we obtain a link from the subset into an objective function value for the total data set. [30] suggested to examine all subsets of size $p+1$, which is infeasible even for small models. We will thus apply TA to this subset selection problem. Algorithms 2.3 and 2.5 remain valid; the neighbourhood is implemented as an exchange of one element from the solution against an index that is currently not in the solution. (A similar approach is taken in [10] for quantile regression.)

We thus need to solve two nested optimisation problems: the outer loop moves through different subsets, while the inner loop needs to find $\hat{\theta}_c$ for the given subset. Fortunately, for a subset of size $p+1$, there exists an exact and fast method to compute the Chebyshev-fit. Let X_s be a subset of X of size $(p+1)\times p$, the corresponding entries of y are stored in the vector y_s. Then Algorithm 2.6 describes a method, based on LS, to obtain the Chebyshev-fit [30, 1].

2.4 Numerical Experiments

All the considered optimisation techniques are stochastic algorithms, so restarting the same algorithm several times for the same data will result in different solutions. We characterise a solution θ by its associated objective function value. We may now

describe the solution obtained from one optimisation run as the realisation of a random variable with an unknown distribution $\mathcal{F}$. For a given data set and a model to estimate (LMS in our case), the shape of $\mathcal{F}$ will depend on the particular optimisation technique, and on the amount of computational resources spent on an optimisation run. Heuristic methods are specially designed such that they can move away from local optima, hence if we allow more iterations, we would expect the method to produce better results on average. In fact, for an ever increasing number of iterations, we would finally expect $\mathcal{F}$ to degenerate to a single point, the global minimum. In practice, we cannot let an algorithm run forever, hence we are interested in the convergence of specific algorithms for finite amounts of computational resources. 'Convergence' here means the change in the shape of $\mathcal{F}$ when we increase the number of iterations. Fortunately, it is straightforward to investigate $\mathcal{F}$: fix the settings for a method (data, parameters, numbers of iterations) and repeatedly restart the algorithm. Thus we obtain a sample of draws from $\mathcal{F}$, from which we can compute an empirical distribution function as an estimate for $\mathcal{F}$.

Since we deal with different heuristics – population-based techniques and trajectory methods – we define computational resources as the number of objective function evaluations. For DE and PS, this is equal to the number of generations times the population size, for TA it is the number thresholds times the steps per threshold. This is justified for LMS regression since the overhead incurred from evolving solutions is small compared with the run time necessary to compute the median of the squared residuals (which requires a least a partial sorting of the squared residuals). Fixing the number of function evaluations has the advantage of allowing us to compare the performance of different methods for a given amount of computational resources. However, we cannot directly compare the subset approach with the direct approach, since in the former the objective function is much more expensive.

We use the experimental setting described in [27], thus we consider the regression model

$$y = X\theta + \epsilon, \tag{2.3}$$

where X is of size $n \times p$, θ is the p-vector of coefficients, and ϵ is Gaussian noise, ie, $\epsilon \sim \mathrm{N}(0,1)$. We always include a constant, so the first column of X is a vector of ones. The remaining elements of X and y are normally distributed with a mean of zero and a variance of one. Thus, the true θ-values are all zero, and the estimated values should be close to zero. We replace, however, about 10% of the observations with outliers. More precisely, if a row in $[y\ X]$ is contaminated with an outlier, it is replaced by

$$[M\ 1\ 100\ 0\ \ldots\ 0]$$

where M is a value between 90 and 200. This setting results in a region of local minima in the search space where θ_2 will be approximately $M/100$. In their paper, [27] analyse how often a given estimator converges to this wrong solution. This analysis, however, confounds two issues: the ability of a given estimator to identify the outliers on the one hand, and the numerical optimisation on the other. Since we are interested in the optimisation, we will not compare coefficients, but look at the value of the objective function.

We set M to 150, and vary the number of regressors p between 2 and 20. The number of observations n is fixed at 400.

2.4.1 Results: Direct Approach

All the methods employed require us to set a number of parameters. We start with 'typical' parameter values: for DE, we set the population size n_P to 10 times the number of coefficients; `CR` and `F` are set to 0.9 and 0.75, respectively. Thus we stay closely with the recommendations of K. Price and R. Storn (see `http://www.icsi.berkeley.edu/~storn/code.html`). For PS, we set n_P to 200, c_1 to 1 and c_2 to 2. Inertia (δ) is set to 1, hence velocity is not reduced systematically. For TA, there are no typical parameter choices, in particular since the neighbourhood function (Algorithm 2.4) is problem-specific. The variable z, which controls the size of the step, was initially set to 0.2.

Figures 2.2, 2.3 and 2.4 give the results for models with 5, 10, and 20 coefficients, respectively. We estimated the distributions (ie, $\mathcal{F}$) by restarting the algorithms 1 000 times. The three panels (top to bottom) in every graphic show the resulting objective function values for 10 000, 20 000, and 30 000 function evaluations.

For the model with 5 coefficients (which is, in practice, a reasonably-sized model), DE gives the best results. With more function evaluations, the DE runs converge on a small number of local minima. The performance of DE deteriorates, though, with more

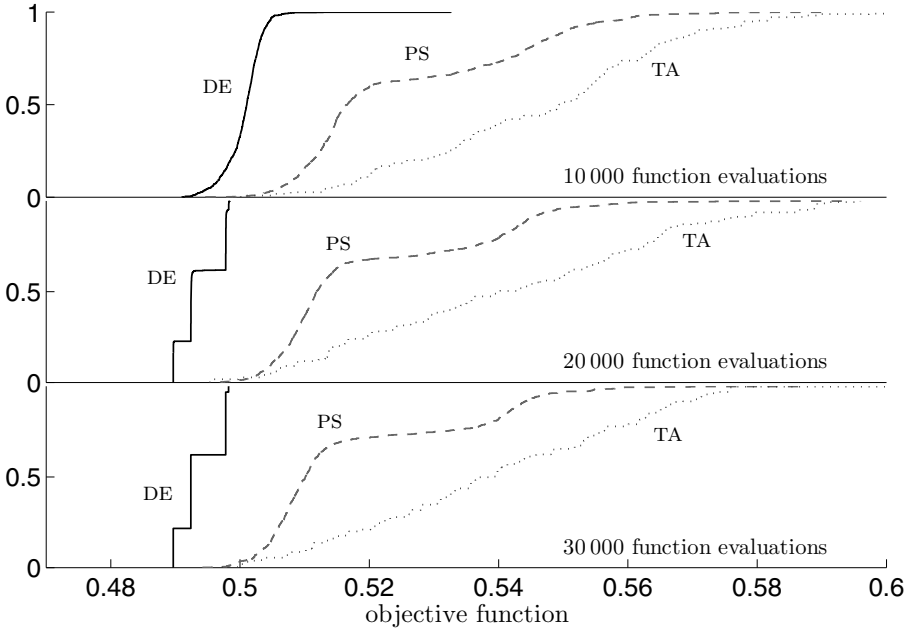


Fig. 2.2. Estimated distributions $\mathcal{F}$: direct approach with $p = 5$.

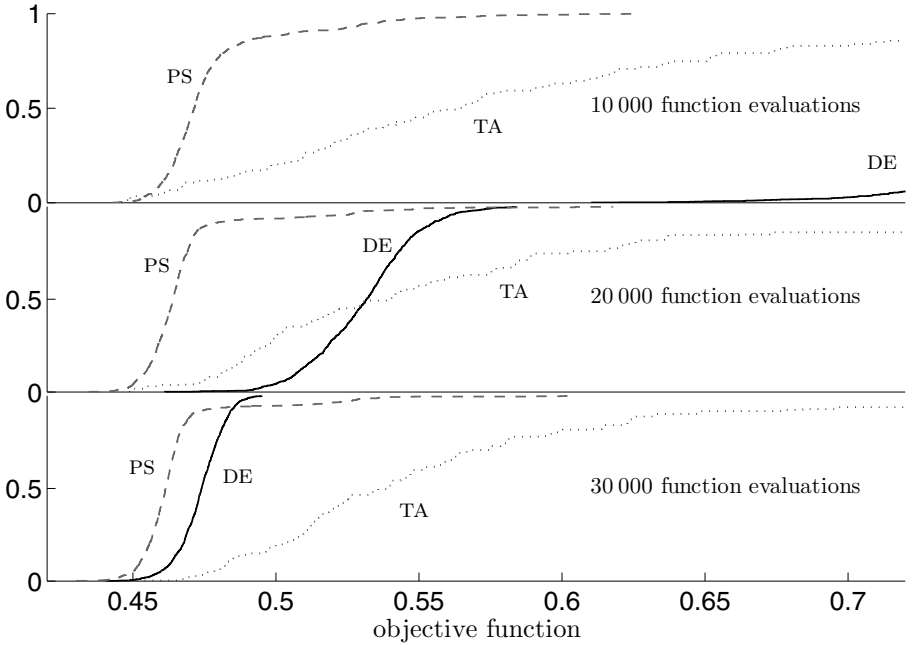


Fig. 2.3. Estimated distributions $\mathcal{F}$: direct approach with $p = 10$.

coefficients, ie, larger models. For $p = 20$ no solution for DE is visible any more in Figure 2.4, the distribution is too far to the right. PS performs best for such larger models, even though the distribution is skewed to the right. In other words, the method occasionally converges on a comparatively bad solution. TA gives reasonable solutions, though generally either DE or PS give better results. In particular, the distribution of solutions for TA is rather dispersed. Take for instance the model with $p = 20$ and 30 000 function evaluations: the probability of reaching the median solution of PS with TA is only about 1%.

These results are conditional on the chosen values for the method's parameters. An important part of implementing heuristics is hence the 'tuning' of the algorithm, ie, finding 'good' parameter values. This search is again an optimisation problem: find those parameter values that lead to optimal (or 'good') results in every restart, ie, parameter values that lead to a 'good' $\mathcal{F}$. Since all methods need several parameters to be set, this optimisation problem is not trivial, in particular since the objective function has to be evaluated from simulation and thus will be noisy. Though this is an (interesting) problem to be investigated, for our purposes here, we do not need such an optimisation – quite the opposite actually. Parameter setting is sometimes portrayed as an advantage, for it allows to adapt methods to different problems. True. But at the same time it requires the analyst who wishes to apply the method to have a much deeper understanding of the respective method. In other words, the analyst will have to be a specialist in optimisation, rather than in finance or econometrics.

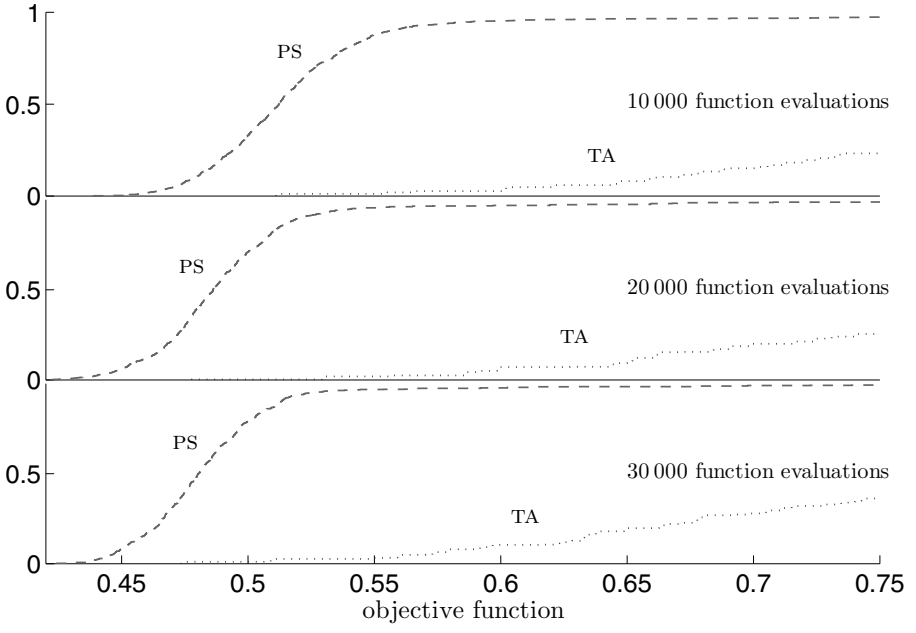


Fig. 2.4. Estimated distributions $\mathcal{F}$: direct approach with $p = 20$.

Boyd and Vandenberghe [3] [p. 5] call a method for solving a particular problem 'a (mature) *technology*, [if it] can be reliably used by many people who do not know, and do not need to know, the details.' (Their example is, fittingly, LS.) If heuristics are to become a technology in this sense, the more pressing question is not whether we have used the 'optimal' parameters, but how sensitive our method's solutions are to specific parameter settings. Since this is a volume on computational finance, let us give a financial analogy: while parameter optimisation may be regarded equivalent to the trading side of a business, we are more interested in risk management.

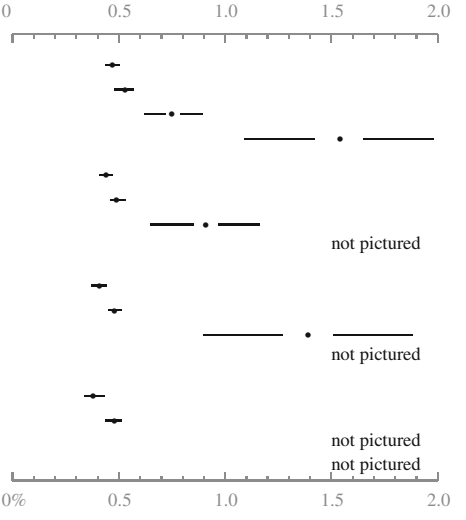
To illustrate this point, we look at the model with $p = 20$ which proved the most difficult, and solve it with different settings for the parameters. The number of function evaluations was set to 30 000. For every parameter setting we conducted 1 000 restarts. All calculations are based on the same data, hence the results in the following tables are directly comparable for different methods.

Parameter Sensitivity for Differential Evolution

Table 2.1 shows the results when we vary `F` and `CR`. We include the median, best, and worst value of the obtained solutions. Furthermore we include quartile plots [31, 12] of the distributions. A quartile plot is constructed like a boxplot, but without the box: it only shows the median (the dot in the middle) and the 'whiskers'.

Table 2.1. Parameter sensitivity DE.

CR	F	median	best	worst
0.2	0.2	0.47	0.424	0.507
	0.4	0.53	0.464	0.575
	0.6	0.75	0.560	0.962
	0.8	1.54	0.988	2.080
0.4	0.2	0.44	0.399	0.472
	0.4	0.49	0.437	0.558
	0.6	0.91	0.631	1.190
	0.8	2.81	1.660	4.030
0.6	0.2	0.41	0.356	0.443
	0.4	0.48	0.410	0.512
	0.6	1.39	0.848	1.880
	0.8	5.36	2.350	7.730
0.8	0.2	0.38	0.338	0.432
	0.4	0.48	0.409	0.523
	0.6	2.29	1.200	3.640
	0.8	9.05	3.360	12.770



The solutions returned by DE improve drastically when we set F to low values while different choices for CR have less influence. This suggests that for LMS-regression, using DE needs to be accompanied by testing of the robustness of the solutions. With small F, we evolve the solutions by adding small changes at several dimensions of the solution. In a sense, then, we have a population of local searches, or at least of slowly-moving individuals.

Parameter Sensitivity for Particle Swarm Optimisation

Tables 2.2–2.5 give the result for PS; here the picture is different. While there are differences in the results for different settings of the parameters, the results are more stable when we vary δ, c_1 and c_2. Each table gives results for different values of c_1 and c_2, with δ fixed for the whole table. The most salient result is that velocity should not be reduced too fast, hence δ should be below but close to one.

Though not reported here, we also reran our initial tests (Figures 2.2, 2.3 and 2.4). With 'improved' parameter values for both DE and PS, both methods performed equally well for small models, but PS still was superior for large models.

Parameter Sensitivity for Threshold Accepting

We ran TA with different values for z (see Algorithm 2.4): 0.05, 0.10, and 0.20. Table 2.6 gives the results. The results indicate that z should be small; in our setting 0.05 performed best on average. At the same time, reducing z deteriorated the worst solution. Thus for too small step sizes, TA more often seemed to get stuck in local, but globally suboptimal, minima.

Table 2.2. Parameter sensitivity PS for $\delta = 1$.

c_2	c_1	median	best	worst
0.5	0.5	0.46	0.384	0.921
	1.0	0.45	0.376	0.944
	1.5	0.45	0.394	0.985
	2.0	0.45	0.399	0.938
1.0	0.5	0.47	0.404	0.872
	1.0	0.46	0.391	0.910
	1.5	0.45	0.371	0.936
	2.0	0.45	0.402	1.030
1.5	0.5	0.46	0.406	0.960
	1.0	0.46	0.395	0.890
	1.5	0.45	0.399	0.926
	2.0	0.45	0.402	0.829
2.0	0.5	0.46	0.402	1.120
	1.0	0.46	0.390	1.010
	1.5	0.45	0.401	0.850
	2.0	0.45	0.392	0.833

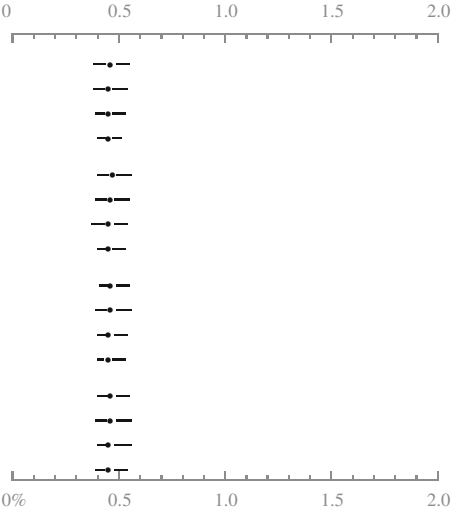
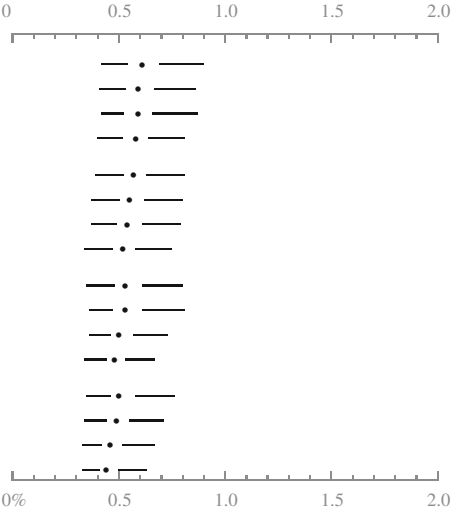


Table 2.3. Parameter sensitivity PS for $\delta = 0.5$.

c_2	c_1	median	best	worst
0.5	0.5	0.61	0.416	1.230
	1.0	0.59	0.409	1.010
	1.5	0.59	0.419	0.935
	2.0	0.58	0.401	0.962
1.0	0.5	0.57	0.385	1.090
	1.0	0.55	0.372	1.040
	1.5	0.54	0.366	0.854
	2.0	0.52	0.343	0.890
1.5	0.5	0.53	0.353	1.030
	1.0	0.53	0.361	1.050
	1.5	0.50	0.360	0.924
	2.0	0.48	0.339	1.070
2.0	0.5	0.50	0.348	0.933
	1.0	0.49	0.337	0.900
	1.5	0.46	0.331	0.867
	2.0	0.44	0.330	0.835



2.4.2 Results: Subset Approach

As a first benchmark for our algorithm we ran a greedy search, described in Algorithm 2.7. That is, for some random initial solution we check all neighbours, and always move to the best one, given it improves the current solution. For any given solution, there are $(p+1)(n-p-1)$ neighbours, hence visiting them all is time-consuming but still feasible. If, at some point, no improvement can be found any more, the search stops.

Table 2.4. Parameter sensitivity PS for $\delta = 0.75$.

c_2	c_1	median	best	worst
0.5	0.5	0.47	0.348	0.890
	1.0	0.46	0.339	0.923
	1.5	0.45	0.339	0.797
	2.0	0.43	0.327	0.806
1.0	0.5	0.46	0.333	0.881
	1.0	0.44	0.324	0.822
	1.5	0.43	0.326	0.810
	2.0	0.41	0.327	0.800
1.5	0.5	0.43	0.328	0.834
	1.0	0.43	0.316	0.818
	1.5	0.42	0.316	0.840
	2.0	0.42	0.338	0.847
2.0	0.5	0.42	0.332	0.818
	1.0	0.42	0.337	0.878
	1.5	0.43	0.327	0.774
	2.0	0.44	0.358	0.873

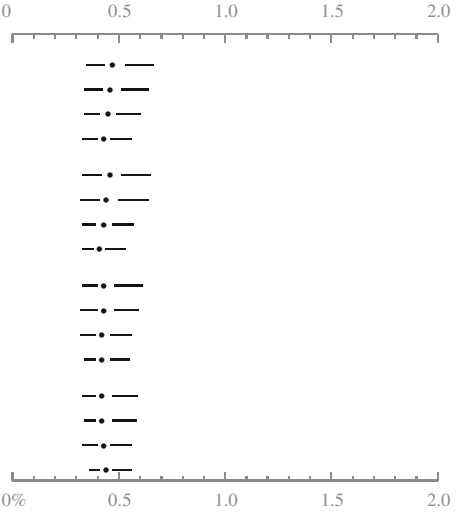


Table 2.5. Parameter sensitivity PS for $\delta = 0.9$.

c_2	c_1	median	best	worst
0.5	0.5	0.41	0.330	0.879
	1.0	0.41	0.328	0.820
	1.5	0.41	0.335	0.776
	2.0	0.42	0.348	0.766
1.0	0.5	0.42	0.335	0.913
	1.0	0.42	0.332	0.884
	1.5	0.42	0.356	0.845
	2.0	0.43	0.365	0.758
1.5	0.5	0.44	0.366	0.882
	1.0	0.44	0.361	0.830
	1.5	0.44	0.367	0.781
	2.0	0.44	0.377	0.832
2.0	0.5	0.45	0.375	0.790
	1.0	0.45	0.386	0.858
	1.5	0.44	0.380	0.922
	2.0	0.44	0.364	0.891

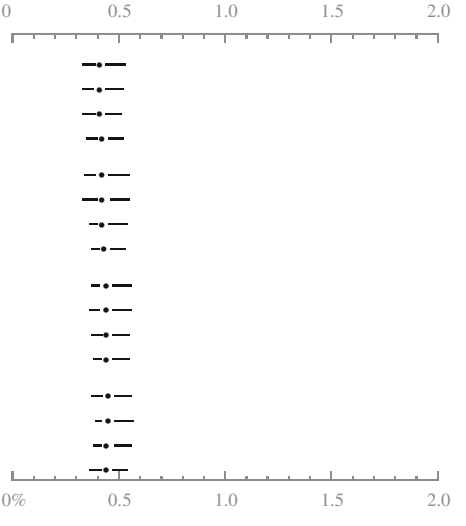
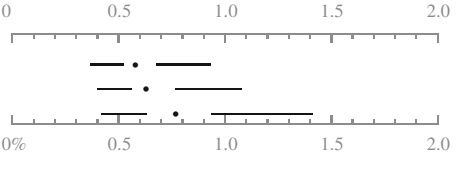


Table 2.6. Parameter sensitivity TA.

z	median	best	worst
0.05	0.58	0.372	6.860
0.10	0.63	0.402	4.840
0.20	0.77	0.425	3.320



Algorithm 2.7. Greedy search for subset selection.

```
select random initial solution θ^c;
set converged = false;
while not converged do
    choose best neighbour θ^best = argmin_{θ^n ∈ N(θ^c)} Φ(θ^n);
    if Φ(θ^best) < Φ(θ^c) then
        θ^c = θ^best;
    end
    converged = true;
end
θ^sol = θ^c;
```

A second benchmark is a classical local search: we start with a random solution and choose a neighbour randomly. If the neighbour is better than the current solution, we move to this new solution. This is equivalent to TA with just one zero-threshold. Results for both searches are shown in Figure 2.5 ($p = 10$), again the distributions are computed from 1 000 restarts. We also add the results for a subset-selection TA with 10 000 function evaluations. Local search performs already much better than the greedy search, and even reaches solutions as good as the TA. The TA runs result in a very steep distribution, thus giving consistently better solutions than the benchmarks.

To illustrate the quality of the solutions obtained with the subset approach, we next plot results for all methods (direct approach and subset approach) for 10 000 function evaluations. It needs to be stressed, though, that the objective function for the subset approach is computationally much more expensive than for the direct approach (one restart needs about 5 times the computing time). We set the parameters of the direct approach techniques to 'good' values (DE: F is 0.2, CR is 0.8; PS: δ is 0.75, c_1 is 2 and c_2 is 1; TA: z is 0.05.) We give just selected results to outline the general findings: with a low level of contamination (10%), for small models, the subset approach gives very good solutions, but lacks behind PS once the model grows. The subset-selection TA is,

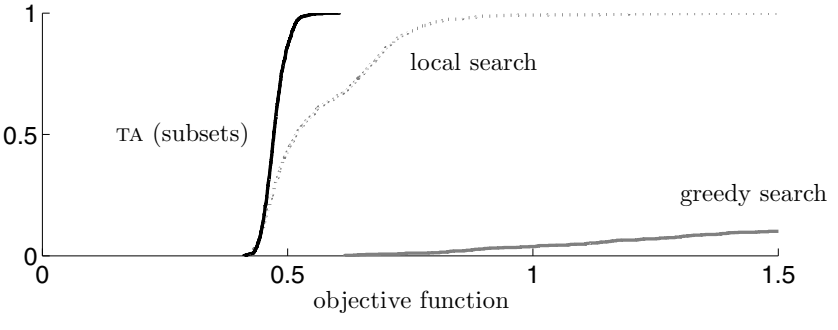


Fig. 2.5. Estimated distributions $\mathcal{F}$: greedy search, local search, and TA (subsets).

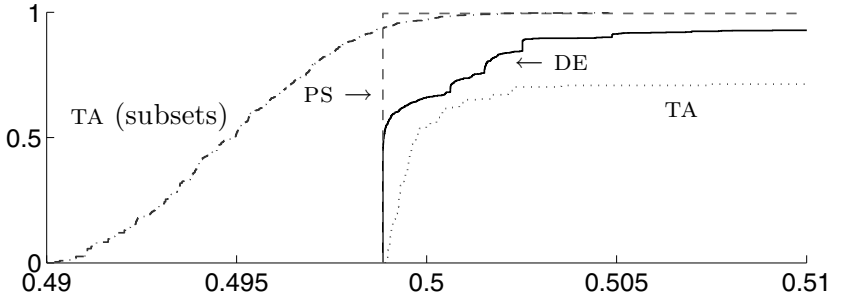


Fig. 2.6. Comparison of $\mathcal{F}$ for models with 10 000 function evaluations ($p = 2$).

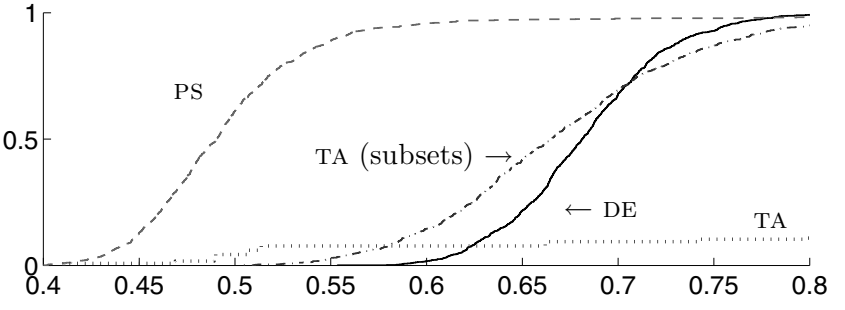


Fig. 2.7. Comparison of $\mathcal{F}$ for models with 10 000 function evaluations ($p = 20$).

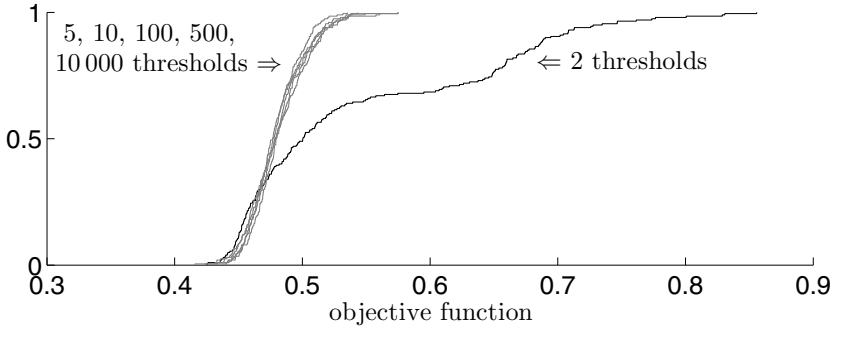


Fig. 2.8. TA (subsets): distributions $\mathcal{F}$ for different numbers of thresholds.

however, very robust when the degree of contamination increases, ie, when the number of outliers increases.

Figure 2.6 and 2.7 show results for $p = 2$ and $p = 20$ with 10% outliers. The distributions are obtained from 1 000 restarts.

Parameter Sensitivity for Threshold Accepting

We ran tests where we fixed the number of function evaluations, but varied the distribution between thresholds (n_{Rounds}) and steps per thresholds (n_{Steps}) (see Algorithm 2.3). Figure 2.8 shows the resulting distributions for 10 000 function evaluations.

The algorithm performs worse for very small numbers of thresholds, but once more than about five thresholds are used, performance becomes stable.

2.5 Conclusion

In this Chapter we described how optimisation heuristics can be used for robust regression. More precisely, we investigated whether Differential Evolution, Particle Swarm Optimisation, and Threshold Accepting are able to minimise the median squared residual of a linear model.

While all the tested methods seem capable of giving 'good' solutions to the LMS-problem, the computational resources (ie, number of function evaluations) would have to be increased drastically to make the distribution of outcomes collapse to a narrow support. In other words, there always remains stochasticity in the solutions. It is difficult to judge the importance of this remaining randomness without a particular application.

For the direct approach we found that while DE performed well for small models, the obtained results were very sensitive to the specific parameter settings once we estimated models with more coefficients. PS showed a much more robust performance. When using good parameter values for both DE and PS, the latter method always dominated DE in our tests. The TA implementations were less efficient in the sense of having much more variable distributions of solutions. The subset approach was more expensive in terms of computing time, but had the advantage of being very robust for different models, in particular for high levels of contamination.

Given its speed and robustness, PS would certainly be our first choice for LMS-estimation. But there are several points to be kept in mind. Firstly, all results are conditional on our model setup. The study of [27] uses one specific data setup; for alternative data the results do not have to be similar. Furthermore, while PS performed well on average, some restarts returned low-quality solutions. It is difficult to judge the relevance of such outcomes: the errors that may occur from the optimisation have to be weighted in light of the actual application, eg, a portfolio construction process. Our suggestion for actual implementations is thus to diversify, that is to implement several methods for the problem given, at least as benchmarks or test cases.

Acknowledgement

The authors gratefully acknowledge financial support from the EU Commission through MRTN-CT-2006-034270 COMISEF. The chapter partially builds on work with Alfio Marazzi whom the authors would like to thank.

A Maximising the Sharpe Ratio

Assume there are p assets, with expected excess returns (over the riskfree rate) collected in a vector $\bar{x}$. The variance–covariance matrix of the assets' returns is Ω. Maximising the Sharpe ratio can be formalised as

$$\max_{\theta} \frac{\theta' \bar{x}}{\sqrt{\theta' \Omega \theta}}.$$

The first-order conditions of this problem lead to the system of linear equations

$$\bar{x} = \Omega\theta,$$

see for instance [7] [ch. 6]. Solving the system and rescaling θ to sum to unity gives the optimal weights.

Assume now that we have T observations; we define $\bar{x}$ to be the sample mean, and collect the p return series in a matrix X of size $T \times p$. For the regression representation as proposed in [4], we need to solve

$$\iota = X\theta^*$$

(which is an LS problem here), where ι is the unit vector and the superscript * only serves to differentiate between θ and θ^*.

This can be rewritten as

$$\begin{aligned} \frac{1}{T} X'\iota &= \frac{1}{T} X'X\theta^*, \\ \bar{x} &= \frac{1}{T} X'X\theta^*, \\ \bar{x} &= \frac{1}{T}(\Omega + \bar{x}\bar{x}')\theta^*. \end{aligned}$$

Applying the Sherman–Morrison formula [16] [ch. 2] allows to show that θ^* will be proportional to θ, and hence after rescaling we have $\theta^* = \theta$.

References

1. Agulló, J.: Exact Algorithms for Computing the Least Median of Squares Estimate in Multiple Linear Regression. In: Dodge, Y. (ed.) L_1-Statistical Procedures and Related Topics. IMS Lecture Notes – Monograph Series, vol. 31, pp. 133–146. IMS (1997)
2. Blume, M.: On the Assessment of Risk. Journal of Finance 26(1), 1–10 (1971)
3. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
4. Britten-Jones, M.: The Sampling Error in Estimates of Mean–Variance Efficient Portfolio Weights. Journal of Finance 54(2), 655–671 (1999)
5. Chan, L., Lakonishok, J.: Robust Measurement of Beta Risk. Journal of Financial and Quantitative Analysis 27(2), 265–282 (1992)
6. Chan, L., Karceski, J., Lakonishok, J.: On Portfolio Optimization: Forecasting Covariances and Choosing the Risk Model. Review of Financial Studies 12(5), 937–974 (1999)

7. Cuthbertson, K., Nitzsche, D.: Quantitative Financial Economics, 2nd edn. Wiley, Chichester (2005)
8. Dueck, G., Scheuer, T.: Threshold Accepting. A General Purpose Optimization Algorithm Superior to Simulated Annealing. Journal of Computational Physics 90(1), 161–175 (1990)
9. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan, pp. 39–43 (1995)
10. Fitzenberger, B., Winker, P.: Improving the computation of censored quantile regressions. Computational Statistics & Data Analysis 52(1), 88–108 (2007)
11. Genton, M., Elvezio Ronchetti, E.: Robust Prediction of Beta. In: Kontoghiorghes, E., Rustem, B., Winker, P. (eds.) Computational Methods in Financial Engineering – Essays in Honour of Manfred Gilli. Springer, Heidelberg (2008)
12. Gilli, M., Schumann, E.: An Empirical Analysis of Alternative Portfolio Selection Criteria. Swiss Finance Institute Research Paper No. 09-06 (2009)
13. Gilli, M., Schumann, E.: Distributed Optimisation of a Portfolio's Omega. Parallel Computing (forthcoming)
14. Manfred Gilli, M., Winker, P.: Heuristic optimization methods in econometrics. In: Belsley, D., Kontoghiorghes, E. (eds.) Handbook of Computational Econometrics. Wiley, Chichester (2009)
15. Gilli, M., Këllezi, E., Hysi, H.: A data-driven optimization heuristic for downside risk minimization. Journal of Risk 8(3), 1–18 (2006)
16. Golub, G., Van Loan, C.: Matrix Computations. John Hopkins University Press, Baltimore (1989)
17. Hyndman, R., Fan, Y.: Sample quantiles in statistical packages. The American Statistician 50(4), 361–365 (1996)
18. Ince, O., Porter, R.B.: Individual Equity Return Data from Thomson Datastream: Handle with Care! Journal of Financial Research 29(4), 463–479 (2006)
19. Kempf, A., Memmel, C.: Estimating the Global Minimum Variance Portfolio. Schmalenbach Business Review 58(4), 332–348 (2006)
20. Klemkosky, R., Martin, J.: The Adjustment of Beta Forecasts. Journal of Finance 30(4), 1123–1128 (1975)
21. Knez, P., Ready, M.: On the Robustness of Size and Book-to-Market in Cross-Sectional Regressions. Journal of Finance 52(4), 1355–1382 (1997)
22. Martin, R.D., Simin, T.: Outlier-Resistant Estimates of Beta. Financial Analysts Journal 59(5), 56–69 (2003)
23. Price, K., Storn, R., Lampinen, J.: Differential Evolution – A practical approach to global optimization. Springer, Heidelberg (2005)
24. Rousseeuw, P.: Least median of squares regression. Journal of the American Statistical Association 79(388), 871–880 (1984)
25. Rousseeuw, P.: Introduction to Positive-Breakdown Methods. In: Maddala, G.S., Rao, C.R. (eds.) Handbook of Statistics, vol. 15, ch. 5. Elsevier, Amsterdam (1997)
26. Rudolf, M., Wolter, H., Zimmermann, H.: A linear model for tracking error minimization. Journal of Banking & Finance 23(1), 85–103 (1999)
27. Salibian-Barrera, M., Yohai, V.: A Fast Algorithm for S-Regression Estimates. Journal of Computational and Graphical Statistics 15(2), 414–427 (2006)
28. Sharpe, W.: Asset Allocation: Management Style and Performance Measurement. Journal of Portfolio Management 18(2), 7–19 (1992)
29. Storn, R., Price, K.: Differential Evolution – a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization 11(4), 341–359 (1997)

30. Stromberg, A.: Computing the Exact Least Median of Squares Estimate and Stability Diagnostics in Multiple Linear Regression. SIAM Journal on Scientific Computing 14(6), 1289–1299 (1993)
31. Tufte, E.: The Visual Display of Quantitative Information, 2nd edn. Graphics Press (2001)
32. Vasicek, O.: A Note on the Cross-Sectional Information in Bayesian Estimation of Security Betas. Journal of Finance 28(5), 1233–1239 (1973)
33. Winker, P.: Optimization Heuristics in Econometrics: Applications of Threshold Accepting. Wiley, Chichester (2001)
34. Winker, P., Fang, K.-T.: Application of threshold-accepting to the evaluation of the discrepancy of a set of points. SIAM Journal on Numerical Analysis 34(5), 2028–2042 (1997)
35. Winker, P., Lyra, M., Sharpe, C.: Least Median of Squares Estimation by Optimization Heuristics with an Application to the CAPM and a Multi Factor Model. Journal of Computational Management Science (2009) (forthcoming)