Robust Least Squares by Iterative Bayesian Data Adjustment

Gilcélia Regiâne de Souza   *Jorge Stolfi*

Technical Report - IC-15-06 - Relatório Técnico

September - 2015 - Setembro

Robust Least Squares by Iterative Bayesian Data Adjustment

Gilcélia Regiâne de Souza[1]
Jorge Stolfi[2]

Abstract

We describe an efficient algorithm for robust multivariate weighted least squares approximation in the presence of outliers. The algorithm iteratively constructs a self-consistent probabilistic interpretation of the data; specifically, the mean and variance of the two populations (inliers and outliers), the least squares fitted approximation, and the probability that each data point is an inlier. An original feature of this algorithm is that, at each iteration, it adjusts the given sampled values according to the estimated probabilities, before re-computing the least squares approximation. This approach is more efficient than the more obvious alternative of including the probabilities in the matrix of the least squares system. The convergence of the method is demonstrated with empirical tests.

1 Introduction

We describe an efficient algorithm for robust multivariate weighted least squares approximation for situations where the sampled function values are a mixture of *inliers* – true values of the measured variable, perturbed by relatively small measurement errors – and *outliers* – spurious values that are not significantly related to the measurement variable. When the least squares method is used in the presence of outliers, these typically pull the fitted function coefficients u away from the true value.

The algorithm iteratively constructs a self-consistent probabilistic interpretation of the data; specifically, the mean and variance of the two populations, the least squares fitted approximation, and the probability that each data point is an inlier. An original feature of this algorithm is that, at each iteration, it adjusts the given sampled values according to the estimated probabilities, before re-computing the least squares approximation. This is more efficient than the more obvious alternative of modifying the data point weights.

This algorithm was described in the Ph. D. Thesis of the first author. [1].

2 Statement of the problem

The problem we address is recovering an unknown linear function F from $\mathbb{R}^m$ to $\mathbb{R}$, given a list of n *observations*. Each observation is a triplet (X_i, f_i, w_i), for $i \in 1..n$ where X_i is a row vector of m real *independent variable samples* X_{ij}, for $j \in 1..n$; f_i is a real *dependent variable sample*; and w_i is a non-negative real *weight* assigned to the observation. The samples f_i

[1]UFSJ, Ouro Branco, Brazil gilcelia@ufsj.edu.br
[2]UNICAMP, Campinas, SP, Brazil stolfi@ic.uniacmp.br

will be viewed as a column n-vector f, and the vectors X_i as rows of a matrix with n rows and m columns.

Each observation (X_i, f_i, w_i) is assumed to be either an inlier or an outlier, but it is not known which. Let κ_i be the (unknown) indicator variable that is 1 if the observation (X_i, f_i, w_i) is an inlier, 0 otherwise.

Let η be the a priori probability of an observation being inlier. For an inlier, the observed value f_i is assumed to be drawn from a Gaussian distribution with an (unknown) standard deviation σ', that is the same for all inliers, and a mean μ'_i that is the (unknown) true value $t_i = F(X_i)$. For an outlier, the measured value f_i is assumed to be drawn from a different Gaussian distribution, whose (unknown) deviation σ'' and (unknown) mean μ'' are the same for all outliers.

The absolute values of the weights w_i are not important, only their relative values. That is, the observations will have the same meaning if all weights w_i are multiplied by the same positive factor. Integer weights w_i can be interpreted as being multiplicities: an observation with integer weight w_i is equivalent to w_i independent observations with weight 1 and same X_i and f_i. Alternatively, the weight w_i can be used to indicate that f_i is drawn from a normal distribution whose deviation is not σ' (resp. σ'') but $\sigma'/\sqrt{w_i}$ (resp. $\sigma''/\sqrt{w_i}$).

The method described here can also be used to fit an affine function of $\mathbb{R}^{m-1}$ to $\mathbb{R}$; it suffices to set all elements in the first column of X to 1.

2.1 Overview of the algorithm

The algorithm tries to find a *probabilistic interpretation* of the data, which we define as comprising (1) explicit assumed values for the parameters η, σ', μ'', and σ''; (2) a column vector n of m *fitted coefficients* for the function F, and (3) an assumed a posteriori probability p_i, with $i \in 1:n$, of observation i being an inlier.

2.2 Self-consistent probabilistic interpretation

This interpretation must be *self-consistent* in the following sense. Let s be the column n-vector of data values predicted by the fitted coefficient m-vector u, namely $s = Xu$.

- The estimated mean of the residuals $f_i - s_i$ for the inliers is zero;
- The estimated variance of the residuals $f_i - s_i$ for the inliers is $(\sigma')^2$;
- The estimated mean of the function samples f_i for the outliers is μ'';
- The estimated variance of the function values f_i for the outliers is $(\sigma'')^2$;
- The probabilities p_i are consistent (in the Bayesian sense) with a Gaussian distribution of inlier residuals $f_i - s_i$ with zero mean and deviation σ', and a Gaussian distribution of for outlier values f_i with mean μ'' and deviation σ'', with prior probability η.
- The prior probability η is consistent with the a posteriori probabilities p_i.
- The vector u is the most likely coefficient vector of the function F, given the above interpretation of the data.

In order to find a self consistent model we compute an initial coefficients vector $u^{(0)}$ by ordinary weighted least squares (see section 2.6.1), and we assign inlier probabilities $p_i = 1/2$ for all observations. Then repeatedly we compute the predicted values s_i, the distribution parameters σ', μ'', σ'' and η, the inlier probabilities p_i, and finally a new vector u by weighted least squares, taking the the probabilities p_i into account (as detailed in Sections 2.6.2–2.6.3).

Empirical evidence (see section 4) shows that this iteration generally converges to a self-consistent model after 3 to 6 iterations, depending on the fraction of outliers and the noise variance of inliers.

2.3 Estimating the distribution parameters

If the indicator variables κ_i were known, the inlier deviation σ' could be estimated by the formula

$$\sigma' = \sqrt{\frac{\sum_i \kappa_i w_i (f_i - s_i)^2}{\sum_i \kappa_i w_i}} \tag{1}$$

where s_i is the value of f_i predicted by the current estimate of the coefficient vector u, that is, $s_i = X_i u$. Since we do not know the indicators κ_i, but only the probabilities $p_i = Pr(\kappa_i = 1)$, we instead estimate that parameter by

$$\sigma' = \sqrt{\frac{\sum_i p_i w_i (f_i - s_i)^2}{\sum_i p_i w_i}} \tag{2}$$

Similarly, the outlier mean μ'' and deviation σ'' are estimated by the formulas

$$\mu'' = \frac{\sum_i (1 - p_i) w_i f_i}{\sum_i (1 - p_i) w_i} \qquad \sigma'' = \sqrt{\frac{\sum_i (1 - p_i) w_i (f_i - \mu'')^2}{\sum_i (1 - p_i) w_i}} \tag{3}$$

2.4 Computing the a priori inlier probability

Assuming chosen the inliner probabilities p_i, the a priori inlier probability η is computed by

$$\eta = \frac{\sum_i p_i w_i}{\sum_i w_i} \tag{4}$$

2.5 Computing the inlier probabilities

Assuming that the parameters σ', μ'', σ'', η and the coefficients vector u are correct, we can compute the probabilities $p_i = Pr(\kappa_i = 1)$. Basically, values of f_i that are far from the current approximation $s_i = X_i u$ are more likely to be outliers.

More precisely, the probability that an observation (X_i, f_i, w_i) is an inlier can be obtained by Bayes's formula,

$$p_i = Pr(\kappa_i = 1 | f_i) = \frac{Pr(f_i | \kappa_i = 1)\eta}{Pr(f_i | \kappa_i = 1)\eta + Pr(f_i | \kappa_i = 0)(1 - \eta)} \tag{5}$$

where

$$Pr(f_i|\kappa_i = 1) = \gamma(f_i - s_i, \sigma')$$

and

$$Pr(f_i|\kappa_i = 0) = \gamma(f_i - \mu'', \sigma'')$$

Here $\gamma(z, \sigma)$ is the Gaussian probability distribution function with mean 0 and deviation σ,

$$\gamma(z, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\Big[-\frac{1}{2}\Big(\frac{z}{\sigma}\Big)^2\Big] \tag{6}$$

Substituting formula (6) into formula (5) we get

$$p_i = \frac{(1/\sigma')\exp\Big[-\frac{1}{2}\Big(\frac{f_i - s_i}{\sigma'}\Big)^2\Big]\eta}{(1/\sigma')\exp\Big[-\frac{1}{2}\Big(\frac{f_i - s_i}{\sigma'}\Big)^2\Big]\eta + (1/\sigma'')\exp\Big[-\frac{1}{2}\Big(\frac{f_i - \mu''}{\sigma''}\Big)^2\Big](1-\eta)} \tag{7}$$

2.6 Fitting the linear function

The coefficient vector u is found by the well-known *weighted least squares* fitting method [2]. Section 2.6.1 reviews the basic method, that would be sufficient if the inliers were precisely known. Section 2.6.2 explains how we use the method given only a probability p_i that each point is an inlier. In Section 2.6.3 we show how to avoid the cost of recomputing the system's matrix A when the probabilities p_i change by modifying the data values f_i.

2.6.1 The weighted least squares method

If the indicators κ_i were known, the most likely candidate for the function F would be given by the least squares method applied to the inliers only. That is, we should seek the coefficient vector $u \in \mathbb{R}^m$ that minimizes the total quadratic error on the inliers,

$$Q(X, w, f, u) = \sum_{i=1}^{n} \kappa_i w_i (X_i u - f_i)^2 \tag{8}$$

The method of least squares can be justified, in statistical terms, as yielding the most likely coefficient vector u given the inlier observations (X_i, f_i, w_i), assuming that the measurement errors included in the values f_i are all normally distributed.

It is well known that the vector u that minimizes formula (8) can be found by requiring the gradient of Q with respect to u to be zero, that is $\partial Q/\partial u_j = 0$ where

$$\frac{\partial Q}{\partial u_j} = \sum_{i=1}^{n} 2\kappa_i w_i (X_i u - f_i) X_{ij} \tag{9}$$

These conditions comprise a system $Au = b$ of n affine equations on n unknowns, where

$$A = X^\top K W X \qquad b = X^\top K W f \tag{10}$$

and K and W are the diagonal matrices with $K_{ii} = \kappa_i$ and $W_{ii} = w_i$ for $i \in 1..n$.

The matrix A is symmetric, and, if the data is sufficient to give a unique solution, it will positive definite. In that case, it has a Cholesky decomposition $A = LL^\top$ where L is lower triangular. The system $Au = b$ then can be solved is two steps, by solving $L^\top v = b$ for v and then $Lu = v$ for u.

2.6.2 Weighted least squares with probabilities

Since the coefficients κ_i are not known, we use instead the expected value of Q, given by

$$\bar{Q}(X, w, f, u) = \sum_{i=1}^{n} p_i w_i (X_i u - f_i)^2 \tag{11}$$

The condition for $\bar{Q}$ to be minimum is then $\partial\bar{Q}/\partial u_j = 0$, where

$$\frac{\partial \bar{Q}}{\partial u_j} = \sum_{i=1}^{n} 2 p_i w_i (X_i u - f_i) X_{ij} \tag{12}$$

The matrices A and b of the linear system $Au = b$ would then be given by

$$A = X^\top P W X \qquad\qquad b = X^\top P W f \tag{13}$$

where P is the diagonal matrix with $P_{ii} = p_i$ for $i \in 1..n$.

2.6.3 Least squares by function value adjustment

If we were to use the least-squares algorithm as described in Section 2.6.2, at every iteration we would have to recompute the matrix A by formula (13) with the new probabilities $P_{ii} = p_i$, which would require time proportional to nm^2; and then recompute its Cholesky factor L, which would require time proportional to m^3.

To save computation time, we instead replace each observed value f_i by an *adjusted value* $\tilde{f}_i$ such that solving the system $\tilde{A}u = \tilde{b}$ with $\tilde{A} = X^\top W X$ and $\tilde{b} = X^\top W \tilde{f}$ has the same result as solving $Au = b$ with $A = X^\top P W X$ and $b = X^\top P W f$. Then the matrix $\tilde{A}$ does not need to be recomputed and factored at each iteration. Namely, we want to replace the derivative formula (12) by a similar formula without the p_i factor, replacing the value f_i by an appropriate adjusted value $\tilde{f}_i$, such that

$$\sum_{i=1}^{n} 2 w_i (X_i u - \tilde{f}_i) X_{ij} = \sum_{i=1}^{n} 2 p_i w_i (X_i u - \tilde{f}_i) X_{ij} \tag{14}$$

which can be obtained by setting

$$\tilde{f}_i = X_i u + p_i (f_i - X_i u) \tag{15}$$

This same approach could be used to efficiently solve several instances of the original problem with the same set of sampling points (that is, the same matrix X) but different weights and observed function values (different vectors w and f).

3 The algorithm

The method described in Section 2.1 is formally specified by the procedure *RobLSQ* below:

Algorithm 1. *RobLSQ*$(X, f, w, k_{\max})$ *returns* (u, p)

```
[$m^2n$]    1.  $A \leftarrow X^\top W X$;
[$m^3$]     2.  $L \leftarrow Cholesky(A)$;
[$mn$]      3.  $b^{(0)} \leftarrow X^\top W f$;
[$m^2$]     4.  $v^{(0)} \leftarrow (L^\top)^{-1} b^{(0)}$;
[$m^2$]     5.  $u^{(0)} \leftarrow L^{-1} v^{(0)}$;
[$n$]       6.  $p^{(0)} \leftarrow (1/2, 1/2, \ldots, 1/2)$;
            7.  for $k$ from 1 to $k_{\max}$ do
[$nm$]          8.  $s^{(k)} \leftarrow X u^{(k-1)}$;
[$n$]           9.  $(\sigma'^{(k)}, \mu''^{(k)}, \sigma''^{(k)}, \eta^{(k)}) \leftarrow Stats(f, s^{(k)}, w, p^{(k-1)})$;
[$n$]           10. $p^{(k)} \leftarrow ProbIn(f, s^{(k)}, \sigma'^{(k)}, \mu''^{(k)}, \sigma''^{(k)}, \eta^{(k)})$;
[$n$]           11. $g^{(k)} \leftarrow Adjust(f, s^{(k)}, p^{(k)})$;
[$nm$]          12. $b^{(k)} \leftarrow X^\top W g^{(k)}$;
[$m^2$]         13. $v^{(k)} \leftarrow (L^\top)^{-1} b^{(k)}$.
[$m^2$]         14. $u^{(k)} \leftarrow L^{-1} v^{(k)}$.
```

The auxiliary procedures used by this algorithm are

- *Cholesky*: Cholesky factorization of square symmetric matrix.
- *Stats*: formulas (2), (3), and (4).
- *ProbIn*: formulas (5), (6), and (7).
- *Adjust*: formula (15).

The algorithm starts by computing, in steps 1–5, an initial coefficient vector $u^{(0)}$ by plain weighted least squares, as if all observations were inliers. In step 6, it assigns arbitrarily the same probability of each data point being an inlier or outlier. Then the algorithm iterates steps 8–14 a preset number $k_{\max}$ of times, to find a self-consistent model of the data — including parameters σ', μ'', σ'' and η, the probabilities p_i, and the coefficient vector u, as described in Secion 2.1. The iteration could be terminated after step 14 if the change in the fitted coefficients vector $u^{(\ell)}$ is small enough.

3.1 Running time

The running time of each step of the *RobLSQ* procedure (Algorithm 1) is proportional to the formula in brackets at the left of the step.

Assuming $n \geq m$, the most expensive steps are step 1, the computation of the matrix A (that requires time proportional to m^2n); and step 2, its Cholesky decomposition takes time proportional to m^3 (proportional to m^3). These steps are executed only once each, so they contribute time $\Theta(m^2 n)$ to the total running time. Solving the two linear systems then takes time proportional to m^2.

Steps 8–14 take $\Theta(mn)$ time per iteration, and are executed up to $k_{\max}$ times. Therefore the total running time will be $\Theta((m+k_{\max})mn)$. The naive approach would require repeating steps 1 and 2 at each iteration, with the extra P matrix as in equation (13). The cost then would be $\Theta(k_{\max} m^2 n)$.

4 Tests

4.1 Data set generation

To test our method, we select values for the numbers m and n of independent and dependent variables, for the number of inliers $n' \leq n$, and for distribution parameters σ'^*, μ''^*, σ''^*. We then generate a random coefficient vector u^* of m elements and a random matrix X with n rows of m elements, both with elements uniformly distributed in the domain $\mathbb{D} = [-1, +1]$. Then we generate a vector κ of n indicators, each independently set to 1 or 0 so that $\sum_i \kappa_i = n'$.

The weight w_i of each observation is computed by the formula

$$w_i = \prod_{j=1}^{m} \sqrt{1 - X_{ij}^2} \tag{16}$$

This formula is 1 when the sampling point X_i is at the center of the domain $\mathbb{D}^m$, and falls to zero at its boundary. The dependent sample vector f is then generated by

$$f_i = \begin{cases} X_i u^* + z'_i & \text{if} \quad \kappa_i = 1 \\ \mu''^* + z''_i & \text{if} \quad \kappa_i = 0 \end{cases}$$

where z'_i and z''_i are normally distributed independent random variables with zero mean and deviations σ'^* and σ''^*, respectively.

4.2 Quality metric

In order to evaluate the progress and accuracy of the *RobLSQ* algorithm, we defined the error metric

$$\epsilon^{(k)} = \sqrt{\frac{\kappa_i w_i (X_i u^{(k)} - X_i u^*)^2}{\sum_i \kappa_i w_i}} \tag{17}$$

that, is the weighted root-mean-square norm of the error $X_i u^{(k)} - X_i u^*$ over the inliers (observations with $\kappa_i = 1$).

4.3 Results

Figures 1–3 show the evolution of the error $\epsilon^{(k)}$ as a function of the iteration number k, for a few test runs with different numbers of outliers and values of σ'^*.

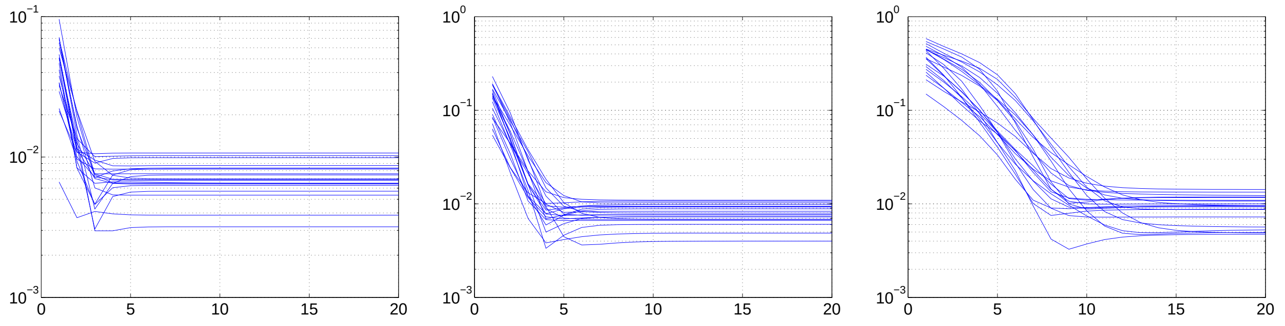

Figure 1: Plot of the error $\epsilon^{(k)}$ as function of the iteration number k, for 20 independent test runs, with parameters $m = 5$, $n = 100$, $\sigma'^{*} = 0.03$, $\mu''^{*} = 0$, and $\sigma''^{*} = 0.18$. The fraction n'/n of outliers in the data was 0.1 (left), 0.2 (middle) and 0.4 (right).

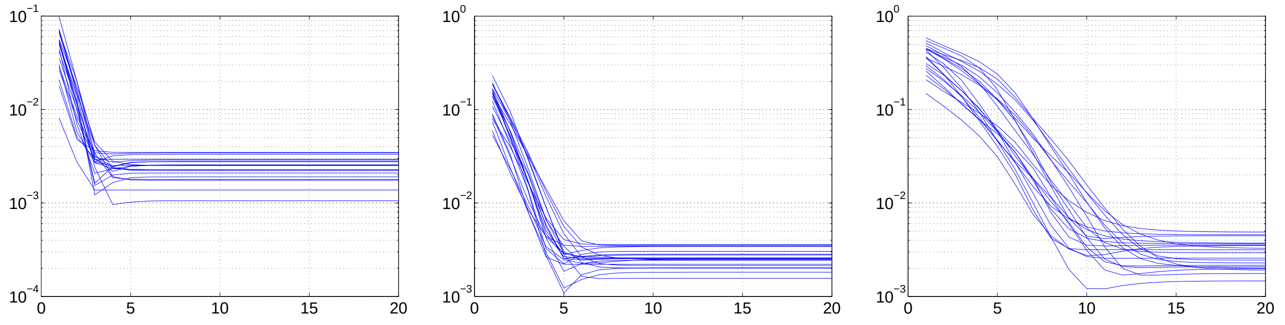

Figure 2: Plot of the error $\epsilon^{(k)}$ as function of the iteration number k, for 20 independent test runs, with parameters $m = 5$, $n = 100$, $\sigma'^{*} = 0.01$, $\mu''^{*} = 0$, and $\sigma''^{*} = 0.18$. The fraction n'/n of outliers in the data was 0.1 (left), 0.2 (middle) and 0.4 (right).

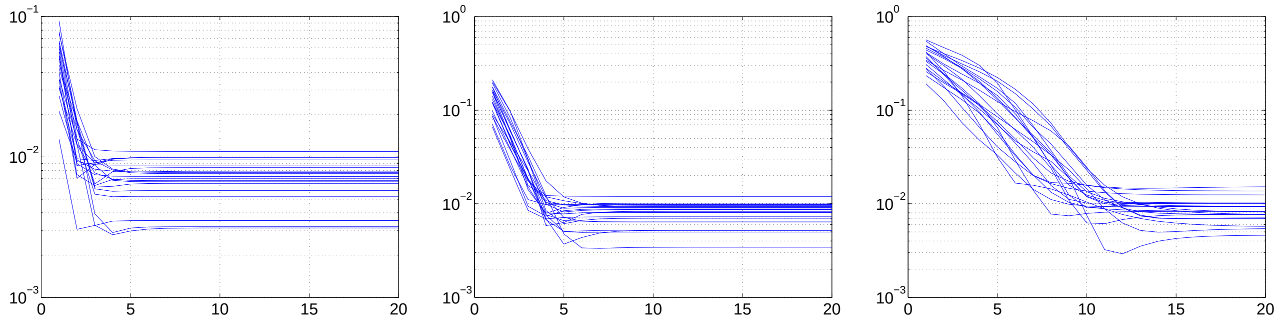

Figure 3: Plot of the error $\epsilon^{(k)}$ as function of the iteration number k, for 20 independent test runs, with parameters $m = 5$, $n = 100$, $\sigma'^{*} = 0.03$, $\mu''^{*} = 0.0$, and $\sigma''^{*} = 0.50$. The fraction n'/n of outliers in the data was 0.1 (left), 0.2 (middle) and 0.4 (right).

5 Conclusions and future work

The procedure *RobLSQ* defined in Section 3 is an efficient iterative method for least squares fitting in the presence of outliers. Tests indicate that the method is robust and converges quickly even when 40% or more of the points are outliers. We believe that the robustness is due to our formulation of the problem as the search for a self consistent statistical model of the data, including Bayesian estimation of the inlier probabilities p_i.

The efficiency of *RobLSQ* is due to our approach of modifying the data values f_i, instead of the effective weights $p_i w_i$, to account for the changes in the probabilities p_i at each iteration. This approach reduces the cost per iteration to $\Theta(mn)$ instead of $\Theta(m^2 n)$. The running time still has an initial term of $\Theta(m^2 n)$; however if the sampling points X_{ij} are shared by several instances of the problem, this cost needs to be paid only once.

The same ideas used in our algorithm could be used to separate observations (X_i, f_i, w_i) that are drawn from a mixture, in unknown proportions, of two or more linear functions of the same m independent variables, each affected by independent errors with a normal distribution with different variance.

References

[1] Gilcélia Regiane de Souza. *Aproximação de Funções Irregularmente Amostradas com Bases Hierárquicas Adaptativas de Elementos Tensoriais Compactos*. PhD thesis, Institute of Mathematics, Statistics and Scientific Computing, UNICAMP, October 2013. Advisor: Jorge Stolfi.

[2] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 1986.