GTSAM Robust Noise Model

Fan Jiang[†], Yetong Zhang[†]

February 2020

1 Introduction

In gtsam, we solve the problem of reducing the error of a factor graph. For each factor i, we have observation function h_i, and the measurement value z_i. Then the measurement error vector e_i is defined as

$$e_i = h_i(x_i) - z_i$$

Then, our objective of reducing the error of the factor graph becomes

$$\min_x err_{graph}(x) = \min_x \sum_i err_i(e_i)$$

Normally, we are concerned with the least square problem, where the error function for each factor is defined as

$$err(e) = \frac{1}{2}\|e\|^2_\Sigma$$

where Σ si the covaraince matrix associated with the measurement. Then, our objective becomes:

$$\min_x \sum_i \frac{1}{2}\|h_i(x_i) - z_i\|^2_{\Sigma_i}$$

However, when outliers exist in the measurements, they may have a large influence on our result. To resolve this issue, we use robust error function instead of the least square error function

$$err(e) = \rho(\|e\|_\Sigma)$$

where ρ is robust loss function. Then, our objective turns into

$$\min_x \sum_i \rho(\|h_i(x_i) - z_i\|_{\Sigma_i})$$

For simplicity, we use m_i to represent the Mahalanobis distance of the measurement error vector: $m_i \doteq \|e_i\|_{\Sigma_i}$, such that each error term becomes $\rho(m_i)$.

Table 1 summarizes the correspondence between the functions used in this document and the functions in gtsam repository

2 Linear Reweighted Least Squares

In [1], linear robust estimation problems were solved with reweighted least squares. In linear cases, the objective is formulated as

$$\sum_i \rho(\|A_i x_i - b_i\|_{\Sigma_i})$$

The objective is minimized by iteratively solving the reweighted least squares problem:

$$\sum_i w(m_i)\|A_i x_i - b_i\|^2_{\Sigma_i}$$

The weight is calculated based on the error's Mahalanobis distance m of the previous iteration

$$w(m) = \frac{\rho'(m)}{m}$$

We can see that in each iteration, a new linear least square problem is created, and solving the problem will generate the weights w for next iteration.

For the linear case, the current gtsam implementation strictly follows this algorithm (with Gauss-Newton optimizer):

Algorithm 1 Linear function with robust noise model

```
set initial value for $x$
while not converge do
    calculate weight for each factor $w(m_i) = \frac{\rho'(m_i(x_i))}{m_i(x_i)}$
    solve the weighted linear LS problem $\min_x \sum_i \frac{1}{2} w(m_i)\|H_i x_i - z_i\|^2_{\Sigma_i}$ with Cholesky or QR factorization
    update $x$ with the optimization result
end
```

3 Nonlinear Reweighted Least Squares

When we turn the robust noise problem from linear to nonlinear, our objective function becomes

$$\sum_i \rho(\|h_i(x_i) - z_i\|_{\Sigma_i}) \tag{1}$$

One way to solve the problem is: Every iteration, we change the objective into weighted nonlinear least square form, and call the nonlinear solver (LM, Dogleg, etc) to solve the problem.

In the current implementation of GTSAM, a faster approach is used. In every iteration, we perform both reweighting and linearization, as in Algorithm 2.

Note, we use A_i, b_i to represent the linearization result of $h_i(x_i) - z_i$ at the linearization point x_0 such that for small Δi

$$h_i(x_0 + \Delta_i) - z_i \approx A_i \Delta_i - b_i \tag{2}$$

Algorithm 2 trust-region method for nonlinear-robust noise problem

```
set initial value for $x$
while not converge do
    calculate Mahalanobis distance of the error for each factor $m_i = \|h_i(x_i) - z_i\|_{\Sigma_i}$
    calculate weight for each factor $w_i = \frac{\rho'(m_i)}{m_i}$
    create a weighted nonlinear LS problem $\sum\limits_i \frac{1}{2} w_i \|h_i(x_i) - z_i\|^2_{\Sigma_i}$
    linearize the problem to $\sum\limits_i \frac{1}{2} w(m_i) \|A_i \Delta_i - b_i\|^2_{\Sigma_i}$
    solve the linearized LS problem
    maintain trust region
    update $x$ with LM/Dogleg rule
end
```

Note that in every iteration, we only needs to solve a linear least square problem by minimizing

$$\sum_i \frac{1}{2} \| \sqrt{w(m_i)} A_i \Delta_i - \sqrt{w(m_i)} b_i \|^2 \tag{3}$$

An interpretation for Algorithm 2 is: we use the weighted linearized least square function (3) as an approximation to our original objective function (1). In the next section, we'll inspect how well the approximation is.

Table 1: function correspondence.

Symbol in Doc	**Class in gtsam**	**Function in gtsam**
$x \mapsto err_{graph}(x)$	NonlinearFactorGraph	error
$x_i \mapsto err(e_i(x_i))$	NonlinearFactor	error
$x_i \mapsto e_i(x_i)$	NonlinearFactor	unwhitenedError
$e_i \mapsto m(e_i)$	noiseModel::Base	mahalanobisDistance
$m \mapsto \rho(m)$	noiseModel::Base	loss
$Eqn2 : \begin{cases} A_i \mapsto \Sigma_i^{-\frac{1}{2}} A_i \\ b_i \mapsto \Sigma_i^{-\frac{1}{2}} b_i \end{cases}$	noiseModel::Gaussian	whiten
$\begin{cases} A_i \mapsto w(m_i) \Sigma_i^{-\frac{1}{2}} A_i \\ b_i \mapsto w(m_i) \Sigma_i^{-\frac{1}{2}} b_i \end{cases}$	noiseModel::Robust	whiten
$m \mapsto \rho(m)$	mEstimator::Base	loss
$m \mapsto w(m)$	mEstimator::Base	weight
$Eqn3 : \begin{cases} A_i \mapsto w(m_i) A_i \\ b_i \mapsto w(m_i) b_i \end{cases}$	mEstimator::Base	reweight

4 Properties of the Weighted Linearized Function

4.1 Function Value

At the linearization point x_0, the value for the original objective function is

$$\rho(m(x_0))$$

while the value for the reweighted linearized function is

$$\frac{1}{2} w(m) m(x_0)^2 = \frac{1}{2} \rho'(m(x_0)) m(x_0)$$

Note: they are not necessarily the same as shown in the example in Section 5.

4.2 Jacobian Value

Interestingly, the Jacobian of the reweighted function agree with the Jacobian of the original function.

Let us define A_u, b_u to be the linearization result of $h(x) - z$ at x_0 by Taylor expansion. e.g.

$$h(x) - z = h(x_0 + \Delta) - z \approx h(x_0) - z + A_u\Delta = A_u\Delta - b_u$$

and we define A_w, b_w to be the whitened A_u, b_u

$$\begin{aligned} A_w &= \Sigma^{-\frac{1}{2}} A_u \\ b_w &= \Sigma^{-\frac{1}{2}} b_u \end{aligned}$$

At the linearization point, the Jacobian for the original function is

$$\begin{aligned} \frac{\partial \rho}{\partial x} =& \frac{\partial \rho}{\partial r}\frac{\partial r}{\partial x} \\ =& \frac{\partial \rho}{\partial r}\frac{\partial r}{\partial (r^2)}\frac{\partial r^2}{\partial x} \\ =& \frac{\rho'(m)}{r}\frac{\partial r^2}{\partial x} \\ =& w(m) A_w^T b_w \end{aligned}$$

whereas the Jacobian of the *reweighted* linearized function is

$$w(m) A_w^T b_w$$

Thus, if we add a constant offset term to the reweighted linearized function to make it align with the original objective function, it is a local approximation, with the accuracy of first order derivative.

5 Example

We use a simple linear 1d function as our example, in this case the robust error function is huber loss with $k = 2$:

$$\rho(m) = \begin{cases} m^2/2 & \text{if } x \leq k \\ k(|m| - k/2) & \text{if } x > k \end{cases}$$

The measurement error function is defined as

$$h(x) - z = x - 2$$

Then, our objective function becomes

$$\rho(x - 2)$$

When linearizing the objective function at point $x_0 = 5$.

The corresponding error plot is shown in Figure 1. Note that the reweighted least square function (blue curve) is not aligned with the original objective function (black curve) at the linearization point $x = 5$. After we add the offset to the reweighted least square function, we get a good local approximation at the linearization point (orange curve).

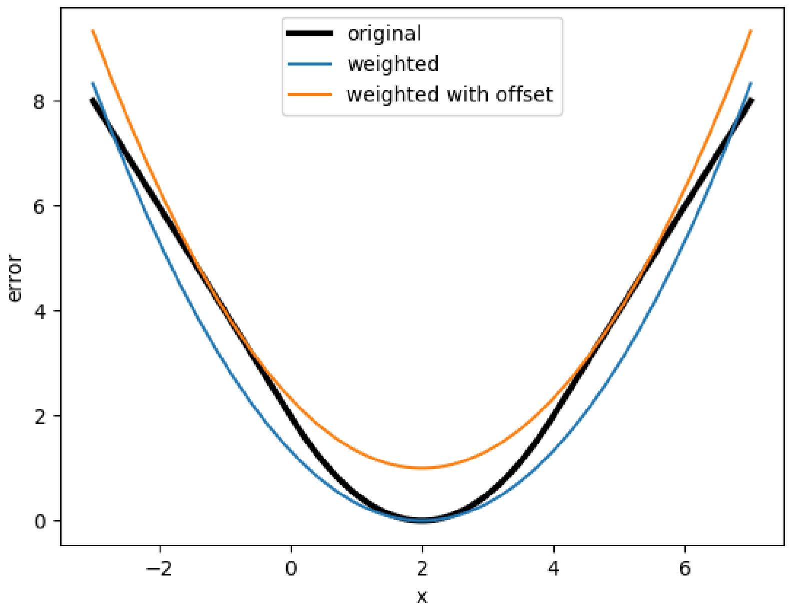

Figure 1: 1D Linear weighted least squares

References

[1] Paul W. Holland and Roy E. Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics - Theory and Methods*, 6(9):813–827, 1977.