Nonlinear Least Squares (NLS)

Sensor Fusion

Fredrik Gustafsson
fredrik.gustafsson@liu.se

Gustaf Hendeby
gustaf.hendeby@liu.se

Linköping University

NLS theory

Nonlinear model, where the term $H_k x$ in the linear model is replaced with $h_k(x)$

$$y_k = h_k(x) + e_k, \quad \mathrm{Cov}(e_k) = R_k, \quad k = 1, \dots, N,$$
$$\mathbf{y} = \mathbf{h}(x) + \mathbf{e}, \quad \mathrm{Cov}(\mathbf{e}) = \mathbf{R}.$$

The NLS solution minimizes

$$\hat{x}^{NLS} = \arg\min_x V^{NLS}(x) = \arg\min_x \frac{1}{2} \sum_{k=1}^{N} (y_k - h_k(x))^T R_k^{-1} (y_k - h_k(x))$$

ML for Gaussian noise with parameter dependent covariance $R(x)$

$$\hat{x}^{ML} = \arg\min_x \left[V^{NLS}(x) + \frac{1}{2} \sum_k \log\det(R_k(x)) \right].$$

Note that we can always evaluate $V(x)$ for a set of grid points $x^{(i)}$ and minimize. This works fine for low-dimensional vectors x ($\dim(x) \leq 3$ as a rule of thumb)

Radar Example

Radar is used as the standard example in Chapter 3. A radar measures bearing φ and range r to an object located at the unknown position $x = (x_1, x_2)^T$.

$$y = \begin{pmatrix} r \\ \varphi \end{pmatrix} = h(x) + e = \begin{pmatrix} \sqrt{x_1^2 + x_2^2} \\ \text{arctan2}(x_1, x_2) \end{pmatrix} + e$$

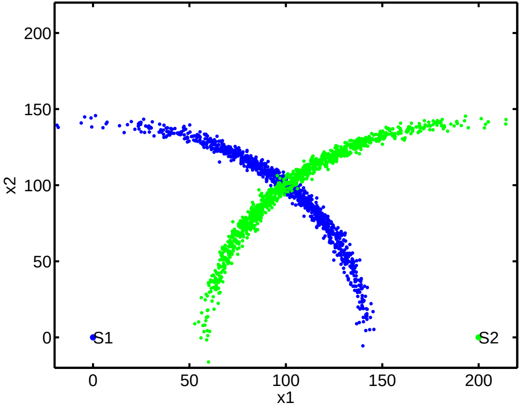
One solution that works in this case is to invert the mapping from x to $y = h(x)$

$$\begin{aligned} x &= h^{-1}(y - e), \\ x_1 &= y_1 \cos(y_2) = (r - e_r)\cos(\varphi - e_\varphi), \\ x_2 &= y_1 \sin(y_2) = (r - e_r)\sin(\varphi - e_\varphi). \end{aligned}$$

It is not so easy to deal with the noise here. This approach is covered by nonlinear transformations, and we leave it for now.

Radar Example

Plot of the mapping $x = h^{-1}(y_k)$ for $N = 100$ measurements of y_k for a target at $x^0 = (100, 100)^T$



How to compute and estimate $\hat{x}$ and covariance $\mathrm{Cov}(\hat{x})$ from this data set? We need a method to consistently over time estimate the position from the radar!

NLS Gradient

Minimize the (unweighted to start with) Nonlinear Least Squares (NLS) cost function

$$V^{NLS}(x) = \frac{1}{2}\sum_{k=1}^{N} \varepsilon_k^T(x)\varepsilon_k(x),$$

A gradient method takes a step in the negative gradient, which here is

$$\frac{dV^{NLS}(x)}{dx} = \sum_{k=1}^{N} \varepsilon_k(x)\frac{d\varepsilon_k(x)}{dx} = J(x)\varepsilon(x)$$

The gradient $J(x)$ will play an important role

$$J(x) = \begin{pmatrix} \frac{\partial \varepsilon_1}{\partial x_1} & \frac{\partial \varepsilon_2}{\partial x_1} & \dots & \frac{\partial \varepsilon_N}{\partial x_1} \\ \frac{\partial \varepsilon_1}{\partial x_2} & \frac{\partial \varepsilon_2}{\partial x_2} & \dots & \frac{\partial \varepsilon_N}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \varepsilon_1}{\partial x_{n_x}} & \frac{\partial \varepsilon_2}{\partial x_{n_x}} & \dots & \frac{\partial \varepsilon_N}{\partial x_{n_x}} \end{pmatrix} = \frac{\partial \varepsilon^T(x)}{\partial x} = -\frac{\partial h^T(x)}{\partial x}.$$

Gradient Search

Simplest numerical algorithm to solve the NLS estimation problem:

Gradient Search

1. Initialize $\hat{x}^{(0)}$ to some decent value.
2. Iterate

$$\hat{x}^{(i+1)} = \hat{x}^{(i)} + \alpha^{(i)} J(\hat{x}^{(i)})\big(\mathbf{y} - \mathbf{h}(\hat{x}^{(i)})\big).$$

   until convergence.

Note:

Here $\alpha^{(i)}$ is a step length that has to be small enough to avoid divergence. It is a scalar that can be optimized in an inner loop to the value that gives the most decrease of the cost function.

Gauss Newton Algorithm

A better and the standard solution to NLS is to apply the Gauss Newton (GN) method. Here the second order derivative of the cost function is needed.

$$\frac{d^2 V^{NLS}(x)}{dx^2} = \sum_{k=1}^{N} \frac{d\varepsilon_k(x)}{dx}\left(\frac{d\varepsilon_k(x)}{dx}\right)^T + \sum_{k=1}^{N} \varepsilon_k(x)\frac{d^2\varepsilon_k(x)}{dx^2}$$

$$= J(x)J^T(x) + \sum_{k=1}^{N} \varepsilon_k(x)\frac{d^2\varepsilon_k(x)}{dx^2}.$$

The GN algorithm neglects the second term in the sum above. Normally, should be much smaller than the first term, which grows with N since it is a quadratic form, which the second term is not.

Gauss-Newton Search

Gauss-Newton Search

1. Initialize $\hat{x}^{(0)}$ to some decent value.
2. Iterate

$$\hat{x}^{(i+1)} = \hat{x}^{(i)} + \alpha^{(i)} \left(J(\hat{x}^{(i)}) J^T(\hat{x}^{(i)}) \right)^{-1} J(\hat{x}^{(i)}) \left(\mathbf{y} - \mathbf{h}(\hat{x}^{(i)}) \right).$$

   until convergence.

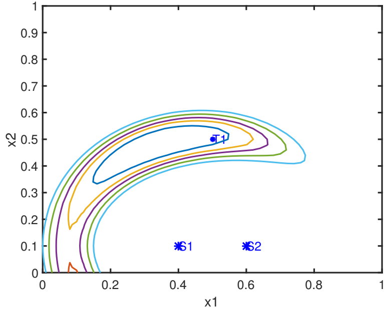
Here $\alpha^{(i)}$ is again a step length that can be optimized in an inner loop.

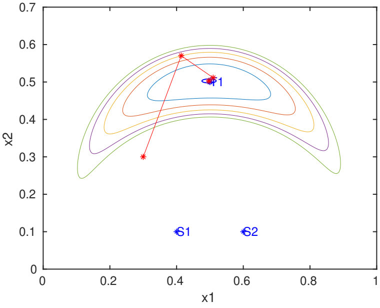
Note:

The only difference between a gradient and the GN search is the modified search direction factor $\left(J(x) J^T(x) \right)^{-1}$.

NLS for TOA

```
th=[0.4 0.1 0.6 0.1]; x0=[0.5 0.5];    % Positions
s=exsensor('toa',2);                   % TOA sensor model
s.th=th; s.x0=x0;                      % Change defaults
s.pe=0.001*eye(2);                     % Noise variance
plot(s), hold on                       % Plot network
y=simulate(s,1);                       % Generate observations
lh2(s,y,[0:0.02:1],[0:0.02:1]);        % Likelihood function plot
```





The likelihood function and the iterations in the NLS estimate.

```
s0=s; s0.x0=[0.3;0.3];                % Prior model for estimation
[xhat,shat,res]=ml(s0,y);             % ML calls NLS
shat                                  % Display estimated signal model
SENSORMOD object: TOA (calibrated from data)
       / sqrt((x(1,:)-th(1)).^2+(x(2,:)-th(2)).^2) \
   y = \ sqrt((x(1,:)-th(3)).^2+(x(2,:)-th(4)).^2) / + e
   x0'  = [0.35,0.49]   + N(0,[0.0091,0.0031;0.0031,0.0017])
   th'  = [0.4,0.1,0.6,0.1]
  States:  x1      x2
  Outputs: y1      y2
xplot2(xhat,'conf',90)                % Estimate and covariance plot
plot(res.TH(1,:),res.TH(2,:),'*-')    % Estimate for each iteration
```

Conditionally linear models

Suppose one part x_l of the parameter vector $x^T = (x_l^T, x_n^T)^T$ appears linearly

$$y_k = h_k(x_n)x_l + e_k, \quad \mathrm{Cov}(e_k) = R_k(x_n),$$

Separable least squares: The WLS solution for x_l is explicitly given by

$$\hat{x}_l^{WLS}(x_n) = \left(\sum_{k=1}^{N} h_k^T(x_n)R_k^{-1}(x_n)h_k(x_n)\right)^{-1} \sum_{k=1}^{N} h_k^T(x_n)R_k^{-1}(x_n)y_k.$$

for each value of x_n.
Plugging in this solution into the nonlinear model, we get a new nonlinear model with less parameters

$$y_k = h_k(x_n)\hat{x}_l^{WLS}(x_n) + \bar{e}_k.$$

Care has to be taken to modify the covariance of the new noise $\bar{e}_k$, which now also includes uncertainty from the estimate besides the measurement noise.

Summary NLS

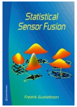
For the nonlinear model $\mathbf{y} = \mathbf{h}(x) + \mathbf{e}, \quad \mathrm{Cov}(\mathbf{e}) = \mathbf{R}$, NLS minimizes

$$\hat{x}^{NLS} = \arg\min_x V^{NLS}(x) = \arg\min_x \frac{1}{2}(\mathbf{y} - \mathbf{h}(x))^T \mathbf{R}^{-1}(\mathbf{y} - \mathbf{h}(x))$$

Approaches to minimize $V^{NLS}(x)$

- Numerically with either a (i) grid search (low-dimensional x, (ii) a gradient method (for almost linear models) or (iii) a Gauss-Newton search (that handles nonlinearities better).
- Further alternatives are provided by nonlinear transforms.

If linear sub-structure exists, $h(x) = h(x_n)x_l$, then x_l can be eliminated with WLS.



Sections 3-3.2 and 3.6