ECE 661: Homework 5 Supplemental[1]

Notation

The nonlinear cost function we wish to minimize is given by

$$C(\mathbf{p}) = \|\mathrm{X} - F(\mathbf{p})\|^2 \tag{1}$$

The error vector $\mathbf{r}(\mathbf{p})$ is given by

$$\mathbf{r}(\mathbf{p}) = \mathrm{X} - F(\mathbf{p})$$

Note that $C(\mathbf{p}) = \mathbf{r}(\mathbf{p})^T\mathbf{r}(\mathbf{p})$

The gradient of a function $y = f(\mathbf{x})$ with $y \in \mathbb{R}$ and $\mathbf{x} \in \mathbb{R}^n$ is given by

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \vdots \\ \frac{\partial y}{\partial x_n} \end{bmatrix}$$

The Jacobian of a function $\mathbf{y} = F(\mathbf{x})$ with $\mathbf{y} \in \mathbb{R}^m$ and $\mathbf{x} \in \mathbb{R}^n$ is given by

$$\mathrm{J}_F(\mathbf{x}) = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

Gradient Descent (GD)

The gradient of a function indicates the direction of maximum increase (i.e. greatest positive slope). In the GD method, we iteratively move in the **opposite** direction of the gradient in small steps until reaching the minimum. At the kth iteration we compute a new set of parameters as follows

$$\mathbf{p}_{k+1} = \mathbf{p}_k - \alpha_k \nabla C(\mathbf{p}_k) \tag{2}$$

The step size $\alpha_k > 0$ is used to control the rate of convergence and can optionally be changed for every iteration. For your implementation you should play around with different values of α_k. The algorithm terminates either when a maximum number of iterations has been reached or when

$$\|\mathbf{p}_{k+1} - \mathbf{p}_k\| < \epsilon$$

for some small ϵ.

Aside: $\nabla C(\mathbf{p}_k) = 2J_{\mathbf{r}}(\mathbf{p}_k)^T\mathbf{r}(\mathbf{p}_k)$ and hence Equation (2) may be rewritten as

$$\mathbf{p}_{k+1} = \mathbf{p}_k - 2\alpha_k J_{\mathbf{r}}(\mathbf{p}_k)^T\mathbf{r}(\mathbf{p}_k) \tag{3}$$

[1]

Prepared by Chad Aeschliman - Purdue University

Gauss-Newton (GN)

While the gradient descent algorithm attempts to reduce the overall cost $C(\mathbf{p})$ at each iteration, the Gauss-Newton method directly attempts to drive the error vector $\mathbf{r}(\mathbf{p})$ to $\mathbf{0}$ (which of course gives the minimum cost as well). A first order approximation of $\mathbf{r}(\mathbf{p}_{k+1})$ is given by

$$\mathbf{r}(\mathbf{p}_{k+1}) \approx \mathbf{r}(\mathbf{p}_k) + J_{\mathbf{r}}(\mathbf{p}_k)\,(\mathbf{p}_{k+1} - \mathbf{p}_k)$$

Setting the right hand side equal to $\mathbf{0}$ and solving for $\mathbf{p}_{k+1}$, we get:

$$\begin{aligned} \mathbf{0} &= \mathbf{r}(\mathbf{p}_k) + J_{\mathbf{r}}(\mathbf{p}_k)\,(\mathbf{p}_{k+1} - \mathbf{p}_k) \\ \therefore\ \mathbf{p}_{k+1} &= \mathbf{p}_k - \left(J_{\mathbf{r}}(\mathbf{p}_k)^T J_{\mathbf{r}}(\mathbf{p}_k)\right)^{-1} J_{\mathbf{r}}(\mathbf{p}_k)^T \mathbf{r}(\mathbf{p}_k) \end{aligned} \tag{4}$$

The criteria for stopping is the same as in the GD method. Since GN directly drives the error vector to $\mathbf{0}$ instead of just the overall cost, GN should converge faster than GD. However, one weakness of this method is that it requires $J_{\mathbf{r}}(\mathbf{p}_k)^T J_{\mathbf{r}}(\mathbf{p}_k)$ to be invertible. The next two methods avoid this constraint.

Levenberg-Marquardt (LM)

The Levenberg-Marquardt algorithm is a hybrid of the GD and GN methods which attempts to combine the advantages of both approaches. At the kth iteration, we compute a new set of parameters as follows:

$$\mathbf{p}_{k+1} = \mathbf{p}_k - \left(J_{\mathbf{r}}(\mathbf{p}_k)^T J_{\mathbf{r}}(\mathbf{p}_k) + \mu_k \mathrm{I}\right)^{-1} J_{\mathbf{r}}(\mathbf{p}_k)^T \mathbf{r}(\mathbf{p}_k)$$

The parameter μ_k controls the nature of the algorithm at a given iteration. If μ_k is close to 0 then LM becomes the same as GN (c.f. Equation (4)). If μ_k is large then LM becomes the same as GD with $\alpha_k = (2\mu_k)^{-1}$ (c.f. Equation (3)).

The problem becomes how to set μ_k at each iteration. For the first iteration we use a heuristic to choose a value, for example

$$\mu_0 = \tau \times \max\left(\mathrm{diag}\left(J_{\mathbf{r}}(\mathbf{p}_0)^T J_{\mathbf{r}}(\mathbf{p}_0)\right)\right)$$

where $0 < \tau \ll 1$. Then, after each iteration we update the value for the next iteration. We first compute the ratio of the actual reduction in the cost function to the expected reduction

$$\rho_{LM} = \frac{C(\mathbf{p}_k) - C(\mathbf{p}_{k+1})}{\delta_{\mathbf{p}}^T \left(\mu_k \delta_{\mathbf{p}} - J_{\mathbf{r}}(\mathbf{p}_k)^T \mathbf{r}(\mathbf{p}_k)\right)} \tag{5}$$

where

$$\delta_{\mathbf{p}} = \mathbf{p}_{k+1} - \mathbf{p}_k \tag{6}$$

If $\rho_{LM} > 0$ then we set

$$\mu_{k+1} = \mu_k \times \max\left(\frac{1}{3}, 1 - (2\rho_{LM} - 1)^3\right)$$

Hence, if the true reduction in cost is close to what we expected then we reduce the value of μ_{k+1} and give more weight to GN.

If $\rho_{LM} \leq 0$ then this iteration actually increased the cost instead of decreasing it! We first undo the damage by setting $\mathbf{p}_{k+1} = \mathbf{p}_k$. We then set

$$\mu_{k+1} = 2\mu_k$$

By increasing the value of μ_{k+1} we give more weight to GD which has safer convergence characteristics.

Dog Leg (DL)

Like the LM method, the dog leg method is a hybrid approach combining GD and GN. At each iteration we first compute two possible values for the parameter vector, one based on GD and the other based on GN

$$\begin{aligned}\mathbf{p}_{GD} &= \mathbf{p}_k - \left(\frac{\|J_{\mathbf{r}}(\mathbf{p}_k)^T\mathbf{r}(\mathbf{p}_k)\|^2}{\|J_{\mathbf{r}}(\mathbf{p}_k)J_{\mathbf{r}}(\mathbf{p}_k)^T\mathbf{r}(\mathbf{p}_k)\|^2}\right) J_{\mathbf{r}}(\mathbf{p}_k)^T\mathbf{r}(\mathbf{p}_k) \\ \mathbf{p}_{GN} &= \mathbf{p}_k - \left(J_{\mathbf{r}}(\mathbf{p}_k)^T J_{\mathbf{r}}(\mathbf{p}_k)\right)^{-1} J_{\mathbf{r}}(\mathbf{p}_k)^T\mathbf{r}(\mathbf{p}_k)\end{aligned}$$

The actual value which is used for $\mathbf{p}_{k+1}$ depends on the size of the *trust region*. The trust region denotes the maximum allowed change in the parameter vector for a given iteration, i.e. we enforce the constraint

$$\|\mathbf{p}_{k+1} - \mathbf{p}_k\| \leq \Delta_k$$

for some parameter Δ_k. The idea is that by limiting how much the parameters can change we remain in a region where the approximations underlying the GD and GN methods are fairly accurate. Let $\delta_{\mathbf{p},GD} = \mathbf{p}_{GD} - \mathbf{p}_k$ and $\delta_{\mathbf{p},GN} = \mathbf{p}_{GN} - \mathbf{p}_k$. Then for a given Δ_k we set $\mathbf{p}_{k+1}$ as follows:

$$\mathbf{p}_{k+1} = \mathbf{p}_k + \begin{cases} \delta_{\mathbf{p},GN} & \|\delta_{\mathbf{p},GN}\| \leq \Delta_k \\ \delta_{\mathbf{p},GD} + \beta(\delta_{\mathbf{p},GN} - \delta_{\mathbf{p},GD}) & \|\delta_{\mathbf{p},GD}\| < \Delta_k < \|\delta_{\mathbf{p},GN}\| \\ \frac{\Delta_k}{\|\delta_{\mathbf{p},GD}\|}\delta_{\mathbf{p},GD} & \text{otherwise} \end{cases} \tag{7}$$

where in the second case, β is chosen s.t. $\|\mathbf{p}_{k+1} - \mathbf{p}_k\| = \Delta_k$. This gives

$$\beta = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

where

$$\begin{aligned} a &= \|\delta_{\mathbf{p},GN} - \delta_{\mathbf{p},GD}\|^2 \\ b &= 2\delta_{\mathbf{p},GD}^T\left(\delta_{\mathbf{p},GN} - \delta_{\mathbf{p},GD}\right) \\ c &= \|\delta_{\mathbf{p},GD}\|^2 - \Delta_k^2 \end{aligned}$$

The motivation behind Equation (7) is to use GN if possible since it gives the fastest convergence (case 1). If this isn't possible (because it falls outside the trust region), we try to use a new value for the parameters which is on the line joining what GN and GD would give (case 2). If this is again not possible we fall back to just using GD, stopping at the perimeter of the trust region (case 3).

The final issue is how to set Δ_k for each iteration. Similar to how μ_k was set for LM, we start with an initial value for the first iteration (try $\Delta_0 = 1$ initially) and then update it after each iteration.

To update Δ_k we first compute the ratio of the actual reduction in the cost function to the expected reduction

$$\rho_{DL} = \frac{C(\mathbf{p}_k) - C(\mathbf{p}_{k+1})}{-\delta_{\mathbf{p}}^T J_{\mathbf{r}}(\mathbf{p}_k)^T J_{\mathbf{r}}(\mathbf{p}_k)\delta_{\mathbf{p}} - 2\mathbf{r}(\mathbf{p}_k)^T J_{\mathbf{r}}(\mathbf{p}_k)\delta_{\mathbf{p}}} \tag{8}$$

where $\delta_{\mathbf{p}}$ is the same as in Equation (6). Regardless of the sign of ρ_{DL}, Δ_{k+1} is given by

$$\Delta_{k+1} = \begin{cases} \frac{1}{4}\Delta_k & \rho_{DL} < \frac{1}{4} \\ \Delta_k & \frac{1}{4} \leq \rho_{DL} \leq \frac{3}{4} \\ 2\Delta_k & \text{otherwise} \end{cases}$$

As with LM, if $\rho_{DL} \leq 0$ then the cost actually increased for this iteration and we undo the damage by setting $\mathbf{p}_{k+1} = \mathbf{p}_k$.