Levenberg-Marquadt
Optimization

• People optimize
  – Stocks
  – Job
  – exam

• Convert qualitative description into quantitative function
  – Objective function
  – Variables
  – constraints
Examples

- Transportation problem
- Chess playing
- Robot path planning
- Computing the optimal shape of an automobile or aircraft
- Controlling a chemical process or a mechanical device to optimize or meet standards of robustness
- Computer Vision
  - Camera Pose estimation
  - Optical flow
  - Stereo depth estimation
  - etc
Optimization

• Minima, maxima or zero of a function
• Local minima vs global minima
Optimization Problems

- Single variable
- Multiple variables
- Linear
- Non-linear
- Unconstrained optimization
- Constraint optimization

\[ \min \left( x_1 - 2 \right)^2 + \left( x_2 - 1 \right)^2 \]

subject to
\[ x_1^2 - x_2 \leq 0, \ x_1 + x_2 \leq 2 \]
Desirable Properties

- Robustness
- Accuracy
- Efficiency
Iterative Solution

- Initial estimate
- Convergence
  - Linear
  - Super linear
  - Quadratic

\[ X^0, X^1, X^2, \ldots X^n \]

\[ X^n \approx X^{n-1} \]

\[ X^n \approx P \]
Rate of Convergence

Definition: Suppose \( \{p_n\}_{n=0}^{\infty} \) is a sequence that converges to \( p \) and that \( e_n = p_n - p \)

\[
\lim_{n \to \infty} \frac{|p_{n+1} - p|}{|p_n - p|^\alpha} = \lim_{n \to \infty} \frac{|e_{n+1}|}{|e_n|^\alpha} = \lambda
\]

then the seq is said to converge to \( p \) of order \( \alpha \) with asymptotic error constant \( \lambda \).

\( \alpha = 1 \), linear
\( \alpha = 2 \), quadratic
\( \alpha = 1 \), and \( \lambda = 0 \), superlinear
Numerical Optimization

• Computation of
  – derivatives,
  – gradient,
  – Jacobian,
  – Hessian

• Analytical derivatives not possible
• Numerical derivatives, finite difference
  – Solution of a linear system (Inverse of a matrix)
Derivative: $f'(x) = \frac{df}{dx}$, $x$ is a scalar

Gradient: $\nabla f(x_1, x_2, \ldots, x_n) = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right)$
Jacobian

\[ F(x_1, x_2, \ldots, x_n) = f_1(x_1, x_2, \ldots, x_n), f_2(x_1, x_2, \ldots, x_n), \ldots, f_m(x_1, x_2, \ldots, x_n) \]

\[ J(F) = \begin{bmatrix}
\frac{\partial f_1}{\partial x_1} & \ldots & \frac{\partial f_1}{\partial x_n} \\
\frac{\partial f_2}{\partial x_1} & \ldots & \frac{\partial f_2}{\partial x_n} \\
\vdots & \ddots & \vdots \\
\frac{\partial f_m}{\partial x_1} & \ldots & \frac{\partial f_m}{\partial x_n}
\end{bmatrix} \]
Hessian

\[ f(x_1, x_2, \ldots, x_n) \]

\[ H(f) = \begin{bmatrix}
\frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\
\frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n}
\end{bmatrix} \]
Optimization Methods

- Gradient Descent
- Conjugate Gradient
- Newton
- Quasi Newton
- Levenberg Marquadt
Weighted Non-linear least squares fit

Consider a set of non-linear equations:

\[ y_i = f(x_i, a) \]

Our aim is to determine a vector such that the following is minimized:

\[ \Psi(a) = \sum_{i=1}^{N} \left( \frac{y_i - f(x_i, a)}{\sigma_i} \right)^2 \]
Newton’s (Inverse Hessian) method:

\[ a_{\text{next}} = a_{\text{current}} + D^{-1}[-\nabla \Psi(a_{\text{current}})] \]  \hspace{1cm} (A)

Where \( D \) is a Hessian matrix

The gradient is given by:

\[
\frac{\partial \Psi}{\partial a_k} = -2 \sum_i \left[ \frac{y_i - f(x_i, a)}{\sigma_i^2} \right] \frac{\partial f(x_i, a)}{\partial a_k} \quad k = 1, \ldots, m
\]

The Hessian is given by:

\[
\frac{\partial^2 \Psi}{\partial a_k \partial a_l} = 2 \sum_{i=1}^N \frac{1}{\sigma_i} \left[ \frac{\partial f(x_i, y)}{\partial a_k} \frac{\partial f(x_i, y)}{\partial a_l} - [y_i - f(x_i, a)] \frac{\partial^2 f(x_i, a)}{\partial a_k \partial a_l} \right] \]

Function to be minimized

\[
\Psi(a) = \sum_{i=1}^N \left( \frac{y_i - f(x_i, a)}{\sigma_i} \right)^2 \quad (D)
\]
Let us define:

\[ \beta_k \equiv -\frac{1}{2} \frac{\partial \Psi}{\partial a_k} \quad \alpha_{kl} \equiv \frac{1}{2} \frac{\partial^2 \Psi}{\partial a_k \partial a_l} \]

Now the Hessian is given by:

\[ \begin{bmatrix} \alpha \end{bmatrix} = \frac{1}{2} D \]

Newton’s method (A) can be written as

\[ a_{\text{next}} = a_{\text{current}} + D^{-1}[-\nabla \Psi(a_{\text{current}})] \quad \text{(A)} \]

Assume

\[ \delta a = a_{\text{next}} - a_{\text{current}} \]
The gradient descent is given by:

\[ a_{\text{next}} = a_{\text{current}} + \text{const}[-\nabla \Psi(a_{\text{current}})] \]  

(C)

\[ \delta a_l = \text{const} \beta_l \quad \delta a = a_{\text{next}} - a_{\text{current}} \]

Assume the second term in (B) is zero:

\[
\frac{\partial^2 \Psi}{\partial a_k \partial a_l} = 2 \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left[ \frac{\partial f(x_i, a)}{\partial a_k} \frac{\partial f(x_i, a)}{\partial a_l} - [y_i - f(x_i, a)] \frac{\partial^2 f(x_i, a)}{\partial a_k \partial a_l} \right]
\]

(B)

\[
\frac{\partial^2 \Psi}{\partial a_k \partial a_l} = 2 \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left[ \frac{\partial f(x_i, a)}{\partial a_k} \frac{\partial f(x_i, a)}{\partial a_l} \right]
\]

Now

\[
\alpha_{kl} = \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left[ \frac{\partial f(x_i, a)}{\partial a_k} \frac{\partial f(x_i, a)}{\partial a_l} \right]
\]
\[ \beta_k \equiv -\frac{1}{2} \frac{\partial \Psi}{\partial a_k} \]
\[ \alpha_{kl} = \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left[ \frac{\partial f(x_i, a)}{\partial a_k} \frac{\partial f(x_i, a)}{\partial a_l} \right] \]

\[ \delta a_l = \text{const } \beta_l \quad (C) \quad \text{Gradient descent} \]

What should be the constant?

The units of \( \beta_k \) are \( 1/a_k \),
therefore units of constant
should be \( a_k^2 \)

Only component of with this property is: \( 1/\alpha_{kk} \)
Let the constant be given by

$$ \text{const} = \frac{1}{\lambda \alpha_{ll}} $$

$$ \delta a_i = \text{const} \beta_i \quad \text{(C)} $$

Gradient descent

Newton from (E)

$$ \lambda \alpha_{ll} \delta a_i = \beta_i \quad \text{(G)} $$

Now define:

$$ \alpha'_{jj} \equiv \alpha_{jj} (1 + \lambda) \quad \text{for } i = j \quad \text{(F)} $$

$$ \alpha'_{kj} \equiv \alpha_{kj} \quad \text{when } j \neq k $$

Combining (E) and (G) and using (F)

$$ \sum_{l=1}^{M} \alpha'_{kl} \delta a_l = \beta_k \quad \text{(H)} $$

L-M
Algorithm

1. Start with some initial estimate of $a$.

2. Compute $\Psi(x_i, a)$ (equation D).  
   \[ \Psi(a) = \sum_{i=1}^{N} \left( \frac{y_i - f(x_i, a)}{\sigma_i} \right)^2 \]

3. Pick a modest value of $\lambda = .001$.

4. Solve linear system (H) for $\delta a$ and evaluate $\Psi(x_i, a + \delta a)$
   \[ \sum_{l=1}^{m} \alpha'_{kl} \delta a_l = \beta_k \]

5. If $\Psi(x_i, a + \delta a) \geq \Psi(x_i, a)$, increase $\lambda$ by a factor of 10, and go to step (4)

6. If $\Psi(x_i, a + \delta a) \leq \Psi(x_i, a)$ decrease $\lambda$ by a factor of 10, update the trial solution: $a \leftarrow a + \delta a$, and go back to step 4.
Szeliski

Projective
Projective (Homographic)

\[
\begin{align*}
x' &= \frac{a_1 x + a_2 y + b_1}{c_1 x + c_2 y + 1} \\
y' &= \frac{a_3 x + a_4 y + b_2}{c_1 x + c_2 y + 1}
\end{align*}
\]
Feature-based Estimation of Homography

\[
x' = \frac{a_1 x + a_2 y + b_1}{c_1 x + c_2 y + 1}
\]

\[
y' = \frac{a_3 x + a_4 y + b_1}{c_1 x + c_2 y + 1}
\]

\[
\begin{bmatrix}
x'_k \\
y'_k
\end{bmatrix} = \begin{bmatrix} x_k & y_k & 1 & 0 & 0 & 0 & -x_k x'_k & -y_k x'_k \\ 0 & 0 & 0 & x_k & y_k & 1 & -x_k y'_k & -y_k y'_k \end{bmatrix} \mathbf{a}
\]

\[
\mathbf{a} = [a_1, a_2, b_1, a_3, a_4, b_2, c_1, c_1]^T
\]
Feature-based Estimation of Homography

\[
\begin{bmatrix}
  x'_1 \\
  y'_1 \\
  x'_k \\
  y'_k
\end{bmatrix}
= \begin{bmatrix}
  x_1 & y_1 & 1 & 0 & 0 & 0 & 0 & -x_1 x'_1 & -y_1 x'_1 \\
  0 & 0 & 0 & x_1 & y_1 & 1 & -x_1 y'_1 & -y_1 y'_1 \\
  x_k & y_k & 1 & 0 & 0 & 0 & 0 & -x_k x'_k & -y_k x'_k \\
  0 & 0 & 0 & x_k & y_k & 1 & -x_k y'_k & -y_k y'_k
\end{bmatrix} a
\]

\[P = Aa\]

Perform least squares fit to compute \(a\).
Szeliski (Image-based estimation of Homography)

\[
x' = \frac{a_1 x + a_2 y + b_1}{c_1 x + c_2 y + 1} \quad \text{Projective}
\]

\[
y' = \frac{a_3 x + a_4 y + b_2}{c_1 x + c_2 y + 1}
\]

\[
E = \sum [f(x', y') - f(x, y)]^2 = \sum e^2
\]

\[
\text{min}
\]
Szeliski

Motion Vector:

\[ \mathbf{m} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & b_1 & b_2 & c_1 & c_2 \end{bmatrix}^T \]
Szeliski (Levenberg-Marquadt)

\[ E = \sum \left[ f(x', y') - f(x, y) \right]^2 = \sum e^2 \]

\[ \alpha_{kl} = \sum_n \frac{\partial e_n}{\partial m_k} \frac{\partial e_n}{\partial m_l} \]

\[ \beta_k = -\sum e \frac{\partial e_n}{\partial m_k} \text{ gradient} \]

\[ \Delta m = (A + \lambda I)^{-1} b \]

Approximation of Hessian \((J^T J, \text{Jacobian})\)
\[ E = \sum \left[ f(x', y') - f(x, y) \right]^2 = \sum e^2 \]

\[ x' = \frac{a_1 x + a_2 y + b_1}{c_1 x + c_2 y + 1} \]

\[ y' = \frac{a_3 x + a_4 y + b_2}{c_1 x + c_2 y + 1} \]

\[ \frac{\partial e}{\partial a_1} = \frac{\partial e}{\partial x'} \frac{\partial x'}{\partial a_1} + \frac{\partial e}{\partial y'} \frac{\partial y'}{\partial a_1} \]
Approximation of Hessian

\[ J^T = \begin{bmatrix} \frac{\partial e_1}{\partial m_1} & \frac{\partial e_2}{\partial m_1} & \cdots & \frac{\partial e_n}{\partial m_1} \\ \frac{\partial e_1}{\partial m_2} & \frac{\partial e_2}{\partial m_2} & \cdots & \frac{\partial e_n}{\partial m_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_1}{\partial m_n} & \frac{\partial e_2}{\partial m_n} & \cdots & \frac{\partial e_n}{\partial m_n} \end{bmatrix} \]

\[ J = \begin{bmatrix} \frac{\partial e_1}{\partial m_1} & \frac{\partial e_1}{\partial m_2} & \frac{\partial e_1}{\partial m_3} & \frac{\partial e_1}{\partial m_4} & \frac{\partial e_1}{\partial m_5} & \frac{\partial e_1}{\partial m_6} & \frac{\partial e_1}{\partial m_7} & \frac{\partial e_1}{\partial m_8} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial e_n}{\partial m_1} & \frac{\partial e_n}{\partial m_2} & \frac{\partial e_n}{\partial m_3} & \frac{\partial e_n}{\partial m_4} & \frac{\partial e_n}{\partial m_5} & \frac{\partial e_n}{\partial m_6} & \frac{\partial e_n}{\partial m_7} & \frac{\partial e_n}{\partial m_8} \end{bmatrix} \]

\[ A = J^T J \]

\[ \alpha_{kl} = \sum_n \frac{\partial e_n}{\partial m_k} \frac{\partial e_n}{\partial m_l} \]

A Matrix
Gradient Vector

\[ b = \left[ -\sum_{n} e_n \frac{\partial e_n}{\partial a_1} \right. \]
\[ \left. -\sum_{n} e_n \frac{\partial e_n}{\partial a_2} \right. \]
\[ \left. -\sum_{n} e_n \frac{\partial e_n}{\partial a_3} \right. \]
\[ \left. -\sum_{n} e_n \frac{\partial e_n}{\partial a_4} \right. \]
\[ \left. -\sum_{n} e \frac{\partial e_n}{\partial b_1} \right. \]
\[ \left. -\sum_{n} e_n \frac{\partial e_n}{\partial b_2} \right. \]
\[ \left. -\sum_{n} e_n \frac{\partial e_n}{\partial c_1} \right. \]
\[ \left. -\sum_{n} e_n \frac{\partial e_n}{\partial c_2} \right. \]

\[ b = \left[ -\sum_{n} e_n \frac{\partial e_n}{\partial m_1} \right. \]
\[ \left. -\sum_{n} e_n \frac{\partial e_n}{\partial m_2} \right. \]
\[ \left. -\sum_{n} e_n \frac{\partial e_n}{\partial m_3} \right. \]
\[ \left. -\sum_{n} e_n \frac{\partial e_n}{\partial m_4} \right. \]
\[ \left. -\sum_{n} e_n \frac{\partial e_n}{\partial m_5} \right. \]
\[ \left. -\sum_{n} e_n \frac{\partial e_n}{\partial m_6} \right. \]
\[ \left. -\sum_{n} e_n \frac{\partial e_n}{\partial m_7} \right. \]
\[ \left. -\sum_{n} e_n \frac{\partial e_n}{\partial m_8} \right. \]
Partial Derivatives wrt motion parameters

\[ \frac{\partial e}{\partial a_1} = \frac{\partial e}{\partial x'} \frac{\partial x'}{\partial a_1} + \frac{\partial e}{\partial y'} \frac{\partial y'}{\partial a_1} \]

\[ x' = \frac{a_1 x + a_2 y + b_1}{c_1 x + c_2 y + 1}, \quad y' = \frac{a_3 x + a_4 y + b_2}{c_1 x + c_2 y + 1} \]

\[ \frac{\partial x'}{\partial a_1} = \frac{x}{c_1 x + c_2 y + 1} \]
\[ \frac{\partial x'}{\partial a_2} = \frac{y}{c_1 x + c_2 y + 1} \]
\[ \frac{\partial x'}{\partial a_3} = 0 \]
\[ \frac{\partial x'}{\partial a_4} = 0 \]
\[ \frac{\partial y'}{\partial a_1} = 0 \]
\[ \frac{\partial y'}{\partial a_2} = 0 \]
\[ \frac{\partial y'}{\partial a_3} = \frac{x}{c_1 x + c_2 y + 1} \]
\[ \frac{\partial y'}{\partial a_4} = \frac{y}{c_1 x + c_2 y + 1} \]
\[ \frac{\partial x'}{\partial b_1} = \frac{1}{c_1 x + c_2 y + 1} \]
\[ \frac{\partial x'}{\partial b_2} = 0 \]
\[ \frac{\partial y'}{\partial b_1} = 0 \]
\[ \frac{\partial y'}{\partial b_2} = \frac{1}{c_1 x + c_2 y + 1} \]
\[ \frac{\partial x'}{\partial c_1} = \frac{-x(a_1 x + a_2 y + b_1)}{(c_1 x + c_2 y + 1)^2} \]
\[ \frac{\partial x'}{\partial c_2} = \frac{-y(a_1 x + a_2 y + b_1)}{(c_1 x + c_2 y + 1)^2} \]
\[ \frac{\partial y'}{\partial c_1} = \frac{-x(a_3 x + a_4 y + b_2)}{(c_1 x + c_2 y + 1)^2} \]
\[ \frac{\partial y'}{\partial c_2} = \frac{-y(a_3 x + a_4 y + b_2)}{(c_1 x + c_2 y + 1)^2} \]
Partial derivatives wrt image coordinates

\[ E = \sum \left[ f(x', y') - f(x, y) \right]^2 = \sum e^2 \]

\[
\frac{\partial e}{\partial x'} = f_x', \\
\frac{\partial e}{\partial y'} = f_y'.
\]
Partial derivatives

\[
\frac{\partial e}{\partial a_1} = \frac{\partial e}{\partial x'} \frac{\partial x'}{\partial a_1} + \frac{\partial e}{\partial y'} \frac{\partial y'}{\partial a_1} = f_x' \frac{x}{c_1 x + c_2 y + 1}
\]
\[
\frac{\partial e}{\partial a_2} = \frac{\partial e}{\partial x'} \frac{\partial x'}{\partial a_2} + \frac{\partial e}{\partial y'} \frac{\partial y'}{\partial a_2} = f_x' \frac{y}{c_1 x + c_2 y + 1}
\]
\[
\frac{\partial e}{\partial a_3} = \frac{\partial e}{\partial x'} \frac{\partial x'}{\partial a_3} + \frac{\partial e}{\partial y'} \frac{\partial y'}{\partial a_3} = f_y' \frac{x}{c_1 x + c_2 y + 1}
\]
\[
\frac{\partial e}{\partial a_4} = \frac{\partial e}{\partial x'} \frac{\partial x'}{\partial a_4} + \frac{\partial e}{\partial y'} \frac{\partial y'}{\partial a_4} = f_y' \frac{y}{c_1 x + c_2 y + 1}
\]
\[
\frac{\partial e}{\partial b_1} = \frac{\partial e}{\partial x'} \frac{\partial x'}{\partial b_1} + \frac{\partial e}{\partial y'} \frac{\partial y'}{\partial b_1} = f_x' \frac{1}{c_1 x + c_2 y + 1}
\]
\[
\frac{\partial e}{\partial b_2} = \frac{\partial e}{\partial x'} \frac{\partial x'}{\partial b_2} + \frac{\partial e}{\partial y'} \frac{\partial y'}{\partial b_2} = f_y' \frac{1}{c_1 x + c_2 y + 1}
\]
\[
\frac{\partial e}{\partial c_1} = \frac{\partial e}{\partial x'} \frac{\partial x'}{\partial c_1} + \frac{\partial e}{\partial y'} \frac{\partial y'}{\partial c_1} = f_x' \frac{-x(a_1 x + a_2 y + b_1)}{(c_1 x + c_2 y + 1)^2} + f_y' \frac{-x(a_3 x + a_4 y + b_2)}{(c_1 x + c_2 y + 1)^2}
\]
\[
\frac{\partial e}{\partial c_2} = \frac{\partial e}{\partial x'} \frac{\partial x'}{\partial c_2} + \frac{\partial e}{\partial y'} \frac{\partial y'}{\partial c_2} = f_x' \frac{-y(a_1 x + a_2 y + b_1)}{(c_1 x + c_2 y + 1)^2} + f_y' \frac{-y(a_3 x + a_4 y + b_2)}{(c_1 x + c_2 y + 1)^2}
\]
Gradient Vector

\[
\mathbf{b} = \begin{bmatrix}
-\sum ef_x' \frac{x}{c_1x + c_2y + 1} \\
-\sum ef_x' \frac{y}{c_1x + c_2y + 1} \\
-\sum ef_y' \frac{x}{c_1x + c_2y + 1} \\
-\sum ef_y' \frac{y}{c_1x + c_2y + 1} \\
-\sum ef_x' \frac{1}{c_1x + c_2y + 1} \\
-\sum ef_y' \frac{1}{c_1x + c_2y + 1}
\end{bmatrix}
\]

\[
\sum_{ex} \left[ \frac{f_x' (a_1x + a_2y + b_1) + f_y' (a_3x + a_4y + b_2)}{(c_1x + c_2y + 1)^2} \right]
\]

\[
\sum_{ey} \left[ \frac{f_x' (a_1x + a_2y + b_1) + f_y' (a_3x + a_4y + b_2)}{(c_1x + c_2y + 1)^2} \right]
\]
Széliski (Levenberg-Marquadt)

- Start with some initial value of $m$, and $\lambda = .001$
  - For each pixel $I$ at $(x_i, y_i)$

- Compute $(x', y')$ using projective transform.

- Compute $e = f(x', y') - f(x, y)$

- Compute $\frac{\partial e}{\partial m_k} = \frac{\partial e}{\partial x'} \frac{\partial x'}{\partial m_k} + \frac{\partial e}{\partial y'} \frac{\partial y'}{\partial m_k}$
Szeliski (Levenberg-Marquadt)

- Compute $A$ and $b$

- Solve system

$$(A - \lambda I) \Delta m = b$$

- Update

$$m^{t+1} = m^t + \Delta m$$
Szeliski (Levenberg-Marquardt)

• check if error has decreased, if not increase $\lambda$ by a factor of 10 and compute a new $\Delta m$

• If error has decreased, decrease $\lambda$ by a factor of 10 and compute a new $\Delta m$

• Continue iteration until error is below threshold.