Data Modeling and Least Squares Fitting 2

COS 323
Nonlinear Least Squares

• Some problems can be rewritten to linear

\[ y = ae^{bx} \]

\[ \Rightarrow (\log y) = (\log a) + bx \]

• Fit data points \((x_i, \log y_i)\) to \(a^* + bx\), \(a = e^{a^*}\)

• Big problem: this no longer minimizes squared error!
Nonlinear Least Squares

- Can write error function, minimize directly

\[ \chi^2 = \sum_i (y_i - f(x_i, a, b, \ldots))^2 \]

Set \( \frac{\partial}{\partial a} = 0, \frac{\partial}{\partial b} = 0 \), etc.

- For the exponential, no analytic solution for \( a, b \):

\[ \chi^2 = \sum_i \left( y_i - ae^{bx_i} \right)^2 \]

\[ \frac{\partial}{\partial a} = \sum_i -2e^{bx_i} \left( y_i - ae^{bx_i} \right) = 0 \]

\[ \frac{\partial}{\partial b} = \sum_i -2ax_ie^{bx_i} \left( y_i - ae^{bx_i} \right) = 0 \]
Newton’s Method

• Apply Newton’s method for minimization:

\[
\begin{bmatrix}
a \\
b \\
\vdots
\end{bmatrix}_{i+1} = \begin{bmatrix}
a \\
b \\
\vdots
\end{bmatrix}_i - H^{-1}G
\]

where \( H \) is Hessian (matrix of all 2\textsuperscript{nd} derivatives) and \( G \) is gradient (vector of all 1\textsuperscript{st} derivatives)
Newton’s Method for Least Squares

\[ \chi^2 = \sum_i (y_i - f(x_i, a, b, \ldots))^2 \]

\[ G = \begin{bmatrix} \frac{\partial^2 (\chi^2)}{\partial a^2} & \frac{\partial^2 (\chi^2)}{\partial a \partial b} & \cdots \\ \frac{\partial^2 (\chi^2)}{\partial a \partial b} & \frac{\partial^2 (\chi^2)}{\partial b^2} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} = \begin{bmatrix} \sum_i -2\frac{\partial f}{\partial a} (y_i - f(x_i, a, b, \ldots)) \\ \sum_i -2\frac{\partial f}{\partial b} (y_i - f(x_i, a, b, \ldots)) \\ \vdots \end{bmatrix} \]

\[ H = \begin{bmatrix} \frac{\partial^2 (\chi^2)}{\partial a^2} & \frac{\partial^2 (\chi^2)}{\partial a \partial b} & \cdots \\ \frac{\partial^2 (\chi^2)}{\partial a \partial b} & \frac{\partial^2 (\chi^2)}{\partial b^2} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \]

• Gradient has 1\textsuperscript{st} derivatives of \( f \), Hessian 2\textsuperscript{nd}
Gauss-Newton Iteration

• Consider 1 term of Hessian:

\[ \frac{\partial^2 (\chi^2)}{\partial a^2} = \frac{\partial}{\partial a} \left( \sum_i \left( y_i - f(x_i, a, b, \ldots) \right)^2 \right) = -2 \sum_i \frac{\partial^2 f}{\partial a^2} (y_i - f(x_i, a, b, \ldots)) + 2 \sum_i \frac{\partial f}{\partial a} \frac{\partial f}{\partial a} \]

• If close to answer, first term close to 0

• Gauss-Newton method: ignore first term!
  
  – Eliminates requirement to calculate 2\textsuperscript{nd} derivatives of \( f \)
Gauss-Newton Iteration

\[
\begin{pmatrix}
   a \\
   b \\
   \vdots
\end{pmatrix}_{i+1} = \begin{pmatrix}
   a \\
   b \\
   \vdots
\end{pmatrix}_i + \delta_i
\]

\[
J_i^T J_i \delta_i = J_i^T r_i
\]

\[
J = \begin{pmatrix}
   \frac{\partial f}{\partial a}(x_1) & \frac{\partial f}{\partial b}(x_1) & \cdots \\
   \frac{\partial f}{\partial a}(x_2) & \frac{\partial f}{\partial b}(x_2) & \cdots \\
   \vdots & \vdots & \ddots
\end{pmatrix}, \quad r = \begin{pmatrix}
   y_1 - f(x_1, a, b, \cdots) \\
   y_2 - f(x_2, a, b, \cdots) \\
   \vdots
\end{pmatrix}
\]

– Surprising fact: still superlinear convergence if “close enough” to answer
Example: Logistic Regression

- Model probability of an event based on values of explanatory variables, using generalized linear model, logistic function $g(z)$

$$p(\bar{x}) = g(ax_1 + bx_2 + \cdots)$$

$$g(z) = \frac{1}{1 + e^{-z}}$$
Logistic Regression

• Uses assumption that positive and negative examples are normally distributed, with different means but same variance

• Applications: predict odds of election victories, sports events, medical outcomes, etc.

• Estimate parameters $a, b, \ldots$ using Gauss-Newton on individual positive, negative examples

• Handy hint: $g'(z) = g(z)(1-g(z))$
Levenberg-Marquardt

- Newton (and Gauss-Newton) work well when close to answer, terribly when far away
- Steepest descent safe when far away
- Levenberg-Marquardt idea: let’s do both

\[
\begin{bmatrix}
  a \\
  b \\
  \vdots
\end{bmatrix}_{i+1} = \begin{bmatrix}
  a \\
  b \\
  \vdots
\end{bmatrix}_i - \alpha G - \beta \begin{bmatrix}
  \frac{\partial f}{\partial a} & \frac{\partial f}{\partial a} \\
  \frac{\partial f}{\partial a} & \frac{\partial f}{\partial b} \\
  \vdots & \vdots
\end{bmatrix}^{-1} G
\]

Steepest descent

Gauss-Newton
Levenberg-Marquardt

- Trade off between constants depending on how far away you are...
- Clever way of doing this:

\[
\begin{pmatrix}
a \\
b \\
\vdots \\
i+1
\end{pmatrix}
= 
\begin{pmatrix}
a \\
b \\
\vdots \\
i
\end{pmatrix}
- 
\begin{pmatrix}
(1 + \lambda) \Sigma \frac{\partial f}{\partial a} \frac{\partial f}{\partial a} \\
\Sigma \frac{\partial f}{\partial a} \frac{\partial f}{\partial b} \\
\vdots
\end{pmatrix}
\begin{pmatrix}
\frac{\partial f}{\partial a} \\
\frac{\partial f}{\partial b} \\
\vdots
\end{pmatrix}
^{-1}
\begin{pmatrix}
\frac{\partial f}{\partial a} \\
\frac{\partial f}{\partial b} \\
\vdots
\end{pmatrix}
\cdot G
\]

- If \( \lambda \) is small, mostly like Gauss-Newton
- If \( \lambda \) is big, matrix becomes mostly diagonal, behaves like steepest descent
Levenberg-Marquardt

• Final bit of cleverness: adjust $\lambda$ depending on how well we’re doing
  – Start with some $\lambda$, e.g. 0.001
  – If last iteration decreased error, accept the step and decrease $\lambda$ to $\lambda/10$
  – If last iteration increased error, reject the step and increase $\lambda$ to $10\lambda$

• Result: fairly stable algorithm, not too painful (no 2nd derivatives), used a lot
Outliers

• A lot of derivations assume Gaussian distribution for errors

• Unfortunately, nature (and experimenters) sometimes don’t cooperate

• Outliers: points with extremely low probability of occurrence (according to Gaussian statistics)

• Can have strong influence on least squares
Robust Estimation

- **Goal:** develop parameter estimation methods insensitive to *small* numbers of *large* errors
- **General approach:** try to give large deviations less weight
- **M-estimators:** minimize some function other than square of $y - f(x,a,b,...)$
Least Absolute Value Fitting

- Minimize $\sum |y_i - f(x_i, a, b, \ldots)|$
  instead of $\sum (y_i - f(x_i, a, b, \ldots))^2$

- Points far away from trend get comparatively less influence
Example: Constant

• For constant function $y = a$, minimizing $\sum(y-a)^2$ gave $a = \text{mean}$
• Minimizing $\sum|y-a|$ gives $a = \text{median}$
Doing Robust Fitting

- In general case, nasty function: discontinuous derivative
- Simplex method often a good choice
Iteratively Reweighted Least Squares

- Sometimes-used approximation: convert to iterated weighted least squares

\[
\sum_i |y_i - f(x_i, a, b, ...)|
\]

\[
= \sum_i \frac{1}{|y_i - f(x_i, a, b, ...)|} (y_i - f(x_i, a, b, ...))^2
\]

\[
= \sum_i w_i (y_i - f(x_i, a, b, ...))^2
\]

with \( w_i \) based on previous iteration
M-Estimators

Different options for weights

- Avoid problems with infinities
- Give even less weight to outliers

\[
\begin{align*}
  w_i &= \frac{1}{|y_i - f(x_i, a, b, \ldots)|} & \text{L}_1 \\
  w_i &= \frac{1}{\varepsilon + |y_i - f(x_i, a, b, \ldots)|} & \text{“Fair”} \\
  w_i &= \frac{1}{\varepsilon + (y_i - f(x_i, a, b, \ldots))^2} & \text{Cauchy / Lorentzian} \\
  w_i &= e^{-k(y_i - f(x_i, a, b, \ldots))^2} & \text{Welsch}
\end{align*}
\]
Iteratively Reweighted Least Squares

• Danger! This is not guaranteed to converge to the right answer!
  – Needs good starting point, which is available if initial least squares estimator is reasonable
  – In general, works OK if few outliers, not too far off
Outlier Detection and Rejection

- Special case of IRWLS: set weight = 0 if outlier, 1 otherwise
- Detecting outliers: \((y_i - f(x_i))^2 > \text{threshold}\)
  - One choice: multiple of mean squared difference
  - Better choice: multiple of median squared difference
  - Can iterate...
  - As before, not guaranteed to do anything reasonable, tends to work OK if only a few outliers
RANSAC

- **RANdom SAmple Consensus**: designed for bad data (in best case, up to 50% outliers)
- Take many random subsets of data
  - Compute least squares fit for each sample
  - See how many points agree: \((y_i - f(x_i))^2 < \text{threshold}\)
  - Threshold user-specified or estimated from more trials
- At end, use fit that agreed with most points
  - Can do one final least squares with all inliers